

320.973
725

В.А.ГВОЗДЕВА

ВВЕДЕНИЕ В СПЕЦИАЛЬНОСТЬ ПРОГРАММИСТА



В. А. Гвоздева

Введение в специальность программиста

*Допущено Министерством образования Российской Федерации
в качестве учебника для студентов образовательных
учреждений среднего профессионального образования
2203 «Программное обеспечение вычислительной техники
и автоматизированных систем»*

НБ ПНУС



687128

Москва
ФОРУМ — ИНФРА-М
2005

УДК 004(075.032)
ББК -018*32.973я723
Г25

Рецензенты:

преподаватель кафедры «Программное обеспечение ВТ и АС»
Московского технического колледжа *Т. Б. Картамышева*;
преподаватель математического колледжа *В. С. Агольцов*

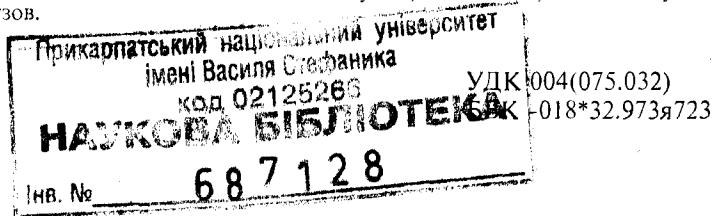
Гвоздева В. А.
Г25 Введение в специальность программиста: Учебник. — М.:
ФОРУМ: ИНФРА-М, 2005. — 208 с.: ил. — (Профессиональное обра-
зование).

ISBN 5-8199-0139-8 (ФОРУМ)
ISBN 5-16-002000-4 (ИНФРА-М)

Цель данной книги — сформировать представление о профессии про-
граммиста и дать основы для приобретения навыков в этой области дея-
тельности.

В учебнике рассматриваются содержание основных понятий програм-
мирования, история его развития, основные элементы и языки програм-
мирования, принципы структурного программирования и начала про-
граммирования в среде Turbo Pascal и системе Delphi. Дается материал о
программном обеспечении для вычислительной техники и автоматизиро-
ванных систем, правовой и программной защите компьютерной инфор-
мации, стандартизации и лицензировании программных продуктов.

Предназначено для учащихся техникумов, колледжей, а также студен-
тов вузов.



ISBN 5-8199-0139-8 (ФОРУМ)
ISBN 5-16-002000-4 (ИНФРА-М)

© В. А. Гвоздева, 2005
© ИД «ФОРУМ», 2005

Введение

Программирование можно рассматривать как искусство, науку, ре-
месло. Программирование — это искусство получения ответов от
машины. Для этого в узком смысле нужно составить специальный
код для технического устройства, а в широком — разработать про-
граммы на языках программирования, т. е. не просто составить код,
а выполнить интеллектуальную работу по составлению высокора-
зумных программ для решения различных задач во всех сферах че-
ловеческой деятельности.

Программирование — процесс описания последовательности
действий решения задачи средствами конкретного языка програм-
мирования и оформление результатов описания в виде программы.
Эта работа требует точности, аккуратности и терпения. Команды
машине должны формулироваться абсолютно четко и полно, не
должны содержать никакой двусмысленности.

На начальном этапе составлением программ для ЭВМ занима-
лись сами изготовители вычислительных машин. Постепенно, с раз-
витием техники, этот процесс из рутинной работы превратился в
интеллектуальную деятельность, сравнимую с искусством, т. к. тру-
доемкое, ручное составление программ было подобно решению
сложных комбинационных задач, которое требовало научных зна-
ний и мастерства. Возникла потребность в людях со специальной
подготовкой и особым складом ума, которых называют программ-
стами. Овладев необходимыми знаниями, научившись грамотно и
творчески применять их в повседневной работе, программист может
стать незаменимым специалистом в своей области деятельности.
Отмечается, что «программист должен обладать способностью пер-
воклассного математика к абстрактному и логическому мышлению
в сочетании с эдисоновским талантом сооружать все что угодно из
0 и 1. Он должен сочетать аккуратность бухгалтера с проницатель-
ностью разведчика, фантазию автора детективов с трезвой практич-
ностью экономиста».

Программист — одна из самых востребованных специальностей
в современном обществе.

С 1970—1980-х гг. программирование как новая научная дисци-
плина занимается методами разработки программных продуктов.

Оно включает комплекс вопросов, связанных с написанием спецификаций, проектированием, кодированием, тестированием и функционированием программ для ЭВМ. Для разработки программного обеспечения применяются следующие методы: математические, инженерных расчетов и управления.

Уровень программирования определяется четырьмя взаимосвязанными факторами развития: возможностями компьютеров, теорией и языками, искусством и технологией программирования.

Профессиональное программирование — вполне прагматичная деятельность, направленная на получение реального программного продукта, которое требует высокой теоретической подготовленности не только в области знания языков программирования и принципов создания программ, но и в области математики, системного анализа, исследования операций, системотехники и др. Программист должен хорошо ориентироваться в уже имеющемся программном обеспечении вычислительной техники и автоматизированных систем, программной защите информации, стандартизации и лицензировании программных продуктов.

Системным программированием, т. е. разработкой средств системного программного обеспечения (ПО) и системы программирования, занимаются системные программисты. Прикладным программированием, т. е. разработкой прикладных программ, занимаются прикладные программисты. Умение хорошо программировать — основное условие успешной профессиональной деятельности программиста. Научиться этому можно, лишь многократно программируя разные задачи, проходя путь от ее постановки до работающей программы.

Для непосредственного решения задач программист должен:

- осознать задачу;
- составить план общего решения;
- выполнить план, т. е. преобразовать его в определенную последовательность действий;
- проверить результат решения, убедиться в его правильности.

Чтобы все это выполнить, специалист должен многое знать и уметь.

Задача данного учебника — дать необходимые знания будущим программистам и приобрести начальные навыки программирования задач в программной среде Turbo Pascal. Получив знания и умения, вы сможете на этой базе осваивать в учебном процессе более трудные разделы программирования и смежных дисциплин, а также начать практическое воплощение своих замыслов, создавая различные программные продукты.

История развития программирования

С глубокой древности известны попытки создать устройства, ускоряющие и облегчающие процесс вычислений. Еще древние греки и римляне применяли приспособление, подобное счетам, — абак. Такие устройства были известны и в странах Древнего Востока. В XVII в. немецкие ученые В. Шиккард (1623), Г. Лейбниц (1673) и французский ученый Б. Паскаль (1642) создали механические вычислительные устройства — предшественники всем известного арифмометра. Вычислительные машины совершенствовались в течение нескольких веков. Но при этом не применялось понятие «программа и программирование».

Только в начале XIX в. (1830) английский ученый, профессор математики Кембриджского университета Чарльз Бэббидж, анализируя результаты обработки переписи населения во Франции, теоретически исследовал процесс выполнения вычислений и обосновал основы архитектуры вычислительной машины. Работая над проектом аналитической машины — «Машины для исчисления разностей», Ч. Бэббидж предсказал многие идеи и принципы организации и работы современных ЭВМ, в частности принцип программного управления и запоминаемой программы. Общая увлеченность наукой дала ученому и Аде Лавлейс (1815—1852) долгие годы плодотворного сотрудничества. В 1843 г. она перевела статью Менабреа по лекциям Ч. Бэббиджа, где в виде подробных комментариев (по объему они превосходили основной текст) сформулировала главные принципы программирования аналитической машины. Она разработала первую программу (1843) для машины Бэббиджа, убедила его в необходимости использования в изобретении двоичной системы счисления вместо десятичной, разработала принципы программирования, предусматривающие повторение одной и той же последовательности команд при определенных условиях. Именно она предложила термины «рабочая ячейка» и «цикл». А. Лавлейс составила первые программы для решения системы двух уравнений и вычисления чисел Бернулли по довольно сложному алгоритму и предположила, что со временем аналитическая машина будет сочинять музыкальные произведения, рисовать картины и использоваться в практической и научной деятельности. Время подтвердило ее правоту и точность прогнозов. Своими работами А. Лавлейс заложила теоретические основы программирования и по праву считается первым в мире программистом и основоположником научного программирования.

В 1854 г. английский математик Джордж Буль опубликовал книгу «Законы мышления», в которой развил алгебру высказываний —

Булеву алгебру. На ее основе в начале 80-х гг. XIX в. построена теория релейно-контактных схем и конструирования сложных дискретных автоматов. Алгебра логики оказала многогранное влияние на развитие вычислительной техники, являясь инструментом разработки и анализа сложных схем, инструментом оптимизации большого числа логических элементов, из многих тысяч которых состоит современная ЭВМ.

Идеи Ч. Бэббиджа реализовал американский ученый Г. Холлерит, который с помощью построенной счетно-аналитической машины и перфокарт за три года обработал результаты переписи населения в США по состоянию на 1890 г. В машине впервые было использовано электричество. В 1896 г. Холлеритом была основана фирма по выпуску вычислительных перфорационных машин и перфокарт.

В 1936 г. английский математик А. Тьюринг ввел понятие машины Тьюринга, как формального уточнения интуитивного понятия алгоритма. Ученый показал, что любой алгоритм в некотором смысле может быть реализован на машине Тьюринга, а следовательно, доказывал возможность построения универсальной ЭВМ. И та, и другая машины аналогично могут быть снабжены исходными данными решаемой задачи и программой ее решения. Машину Тьюринга можно считать как бы идеализированной моделью универсальной ЭВМ.

В 40-х гг. XX в. механическая элементная база вычислительных машин стала заменяться электрическими и электронными устройствами. Первые электромеханические машины были созданы в Германии К. Цузе (Ц-3, 1941 г.) и в США под руководством профессора Гарвардского университета Г. Айкена (МАРК-1, 1944 г.). Первая электронная машина создана в США группой инженеров под руководством доктора Пенсильванского университета Дж. Мочли и аспиранта Дж. Эккерта (ЭНИАК — электронный числовой интегратор и калькулятор, 1946 г.). В 1949 г. в Англии была построена EDSAC — первая машина, обладающая автоматическим программным управлением, внутренним запоминающим устройством и другими необходимыми компонентами современных ЭВМ.

Логические схемы вычислительных машин были разработаны в конце 1940-х гг. Дж. фон Нейманом, Г. Гольдштейном и А. В. Берксом. Особый вклад в эту работу внес американский математик Джон фон Нейман, принимавший участие в создании ЭНИАК. Он предложил идею хранения команд управления и данных в машинной памяти и сформулировал основные принципы построения со-

временных ЭВМ. ЭВМ с хранимой программой оказались более быстроедействующими и гибкими, чем ранее созданные.

В 1951 г. в США было налажено первое серийное производство электронных машин УНИВАК (универсальная автоматическая вычислительная машина). В это же время фирма IBM начала серийный выпуск машины IBM/701.

В СССР первыми авторами ЭВМ, изобретенной в декабре 1948 г., являются И. С. Брук и Б. И. Рамеев. А первая советская ЭВМ с сохраняющейся программой создана в 1951 г. под руководством С. А. Лебедева (МЭСМ — малая электронная счетная машина). В 1953 г. в Советском Союзе начался серийный выпуск машин, первыми из которых были БЭСМ-1, «Стрела».

С появлением цифровых программно-управляемых машин родилась новая область прикладной математики — программирование. Как область науки и профессии она возникла в 1950-х гг. Первоначально программы составлялись вручную на машинных языках (в машинных кодах). Программы были громоздки, их отладка — очень трудоемка. Для упрощения приемов и методов составления и отладки программ были созданы мнемокоды, по структуре близкие к машинному языку и использующие символьную адресацию. Ассемблеры переводили программу, записанную в мнемокоде, на машинный язык и, расширенные макрокомандами, используются и в настоящее время. Далее были созданы автокоды, которые можно применять на различных машинах, и позволившие обмениваться программами. Автокод — набор псевдокоманд для решения специализированных задач, например научных или инженерных. Для таких задач имеется развитая библиотека стандартных программ.

До конца 1950-х гг. ЭВМ основным элементом конструкции были электронные лампы (1-е поколение). В этот период развитие идеологии и техники программирования шло за счет достижений американских ученых Дж. фон Неймана, сформулировавшего основные принципы построения ЭВМ, и Дж. Бэкуса, под руководством которого в 1954 г. был создан Fortran (Formula Translation) — первый язык программирования высокого уровня, используемый до настоящего времени в разных модификациях. Так, в 1965 г. в Дартмутском колледже Д. Кэмэни и Т. Куртцем была разработана упрощенная версия Фортрана — Basic. В 1966 г. комиссия при Американской ассоциации стандартов (ASA) разработала два стандарта языка: Фортран и Базисный Фортран. Используются также дальнейшие модификации языка (например 1970, 1990 гг.).

Достижения в области электроники и микроэлектроники позволили заменить элементную базу ЭВМ на более совершенную.

В конце 1950-х гг. громоздкие электронные лампы заменяют полупроводниками (миниатюрными транзисторами). Появляются ЭВМ II поколения; затем примерно через 10 лет — ЭВМ III поколения на интегральных схемах; еще через 10 лет — ЭВМ IV поколения на больших интегральных схемах (БИС). В Японии в 1990-х гг. реализованы проекты ЭВМ V поколения, в которых использованы достижения в области искусственного интеллекта и биоэлектроники. Если объем оперативного запоминающего устройства (ОЗУ) одной из лучших отечественных машин 1960-х гг. М-20, созданной под руководством С. А. Лебедева в 1958 г., имел 4096 слов (8 Кбайт) и быстродействие 20 тыс. операций в секунду, то современные персональные компьютеры характеризуются ОЗУ в десятки Мбайт и быстродействием в сотни миллионов операций в секунду, что позволяет решать сложнейшие задачи.

В 1953 г. А. А. Ляпуновым был предложен операторный метод программирования, который заключался в автоматизации программирования, а алгоритм решения задачи представлялся в виде совокупности операторов, образующих логическую схему задачи. Схемы позволяли расчленить громоздкий процесс составления программы, части которой составлялись по формальным правилам, а затем объединялись в целое. Для проверки идей операторного метода в СССР в 1954 г. была разработана первая программирующая программа ПП-1, а в 1955 г. более совершенная — ПП-2. В 1956 г. разработана ПП БЭСМ, в 1957 г. — ППСВ, в 1958 г. — для машины «Стрела».

В США в 1954 г. стал применяться алгебраический подход, совпадающий, по существу, с операторным методом. В 1956 г. корпорацией IBM разработана универсальная ПП Фортран для автоматического программирования на ЭВМ IBM/704.

В этот период по мере накопления опыта и теоретического осмысления совершенствовались языки программирования. В 1958—1960 гг. в Европе был создан ALGOL, который породил целую серию алголоподобных языков: Algol W, (1967), Algol 68, Pascal (Н. Вирт, 1970 г.), C (Д. Ритчи и Б. Керниган, 1972 г.), Ada (под руководством Ж. Ишбиа, 1979 г.), C++ (1983). В 1961—1962 гг. Дж. Маккарти в Массачусетском технологическом институте был создан язык функционального программирования Lisp, открывший в программировании одно из альтернативных направлений, предложенных Дж. фон Нейманом.

На начало 1970-х гг. существовало более 700 языков высокого уровня и около 300 трансляторов для автоматизации программирования.

Усложнение структуры ЭВМ привело (в 1953 г. для машин II-го поколения) к созданию операционных систем (ОС) — специальных управляющих программ для организации и решения задач на ЭВМ. Например, мониторинговая система МТИ, созданная в Массачусетском технологическом институте, обеспечивала пакетную обработку, т. е. непрерывное, последовательное прохождение через ЭВМ многих групп (пакетов) заданий и пользование библиотекой служебных программ, хранимой в машине. Это позволило совместить операции по запуску с выполнением программ.

Для ПЭВМ к настоящему времени разработаны ОС: MS DOS, Windows, OS/2, MacOS, Unix, Linux и др. Широкое распространение получили ОС MS DOS и Windows, имеющие развитый интерфейс и широкий набор приложений, позволяющих последовательное выполнение заданий из пакета, обработку различной информации во многих сферах человеческой деятельности.

В 1965 г. итальянцы Бом и Джакопини предложили использовать в качестве базовых алгоритмических элементов следование, ветвление и цикл. Почти в то же время к аналогичным выводам пришел голландский ученый Э. Дейкстра, заложивший основы структурного программирования. В 1970-х гг. эта методология оформилась, и корпорация IBM сообщила о применении в разработке программного обеспечения «Усовершенствованных методов программирования», одним из компонентов которых являлась технология нисходящего структурного программирования (*структурного программирования*), основу которого составляет следующее:

- сложная задача разбивается на простые, функционально управляемые задачи, каждая задача имеет один вход и один выход; управляющий поток программы состоит из совокупности элементарных функциональных подзадач;
- управляющие структуры просты, т. е. логическая задача должна состоять из минимальной, функционально полной совокупности достаточно простых управляющих структур;
- программа разрабатывается поэтапно, на каждом этапе решается ограниченное число точно поставленных задач.

Четко сформулированные основы нисходящей разработки, структурного кодирования и сквозного контроля позволяли перейти к промышленным методам разработки программного обеспечения.

Развитие получило *модульное программирование*, основа которого заключается в следующем:

- функциональная декомпозиция (разбиение) задачи на самостоятельные подзадачи — модули, связанные только входными и выходными данными;

- модуль представляет собой «черный ящик», позволяющий разрабатывать части программ одного проекта на разных языках программирования, а затем с помощью компоновочных средств объединять их в единый загрузочный модуль;
- должно быть ясное понимание назначения всех модулей задачи и их оптимального сочетания;
- с помощью комментариев должно описываться назначение всех переменных модуля.

В период 1970—1980-х гг. развитие теоретических исследований оформило программирование как самостоятельную научную дисциплину, занимающуюся методами разработки программного обеспечения (ПО).

В истории развития промышленного программирования большую роль сыграл программист и бизнесмен Билл Гейтс (Gates William Henry, р. в 1955 г.). Его история очень поучительна для начинающих программистов. В 1972 г. Билл Гейтс и его школьный товарищ Пол Аллен основали компанию по анализу уличного движения «Трэф-О-Дейта» и использовали для обработки данных компьютеры с микропроцессором 8008 — первым из знаменитого ряда микропроцессоров компании «Intel». Будучи студентом Гарвардского университета, в 1975 г. он совместно с Алленом написал для компьютера Altair (фирмы MITS) интерпретатор — программу-переводчик с языка программирования на язык машинных кодов. Они заключили с владельцем фирмы соглашение, по которому их программы распространялись вместе с компьютерами. Товарищи основали компанию «Microsoft», в которой Б. Гейтсу принадлежало 60 % акций, П. Аллену — 40 %. В 1976 г. Гейтс ввел в практику продажу лицензий на свои программные продукты непосредственно производителям компьютеров, что позволило «встраивать» их (ОС и трансляторы с языков программирования) в компьютеры. Это было большое достижение в области маркетинга, принесшее фирме огромные доходы. Фирма привлекала таких новых заказчиков, как фирмы «Apple», «Commodor», «Tendi». В 1980 г. фирма IBM предложила «Microsoft», в которой тогда работало около двух десятков человек, создать языки программирования для ее нового персонального компьютера, в дальнейшем известном как IBM PC. В 1981 г. «Microsoft» приобрела у разработчика Т. Патерсона дисковую ОС (DOS), и в августе этого года IBM PC поставлялась вместе с ОС MS DOS. Успех был настолько велик, что, кроме значительных доходов, привел к тому, что и архитектура Intel, и компьютеры IBM, и программы «Microsoft» фактически стали отраслевыми стандартами. В 1988 г. «Microsoft» создала свою ОС Windows с мощным графиче-

ским интерфейсом. К 1995 г. ОС, выпускаемые фирмой, использовали 85 % персональных компьютеров. ОС Windows совершенствуется год от года, обладая уже средствами доступа в глобальную сеть Internet. Вместе с фирмой NBC был создан круглосуточный кабельный информационный канал новостей. Совместно с фирмой «Энкарта» создана мультимедиа-энциклопедия на CD-ROM «Книжная полка», содержащая электронные версии семи больших справочников, электронную энциклопедию кино — «Синемания». В 1995 г. в фирме «Microsoft» работало 18 тыс. человек, годовой выпуск достиг 200 программных продуктов, а доходы составили миллиарды долларов. В 1998 г. Б. Гейтс стал самым богатым человеком в мире, а в конце 1999 г. — объявил о своем решении уйти с поста главы компании и заняться программированием.

Профессиональное программирование вышло на уровень технологии. Методы разработки ПО синтезируют:

- методы инженерных расчетов для оценки затрат и выбора решений;
- математические методы для составления алгоритмов;
- методы управления для определения требований к системе, учета ситуаций, организации работ и прогнозирования.

На смену структурному программированию в начале 1990-х гг. пришло *объектно-ориентированное программирование* — ООП. Его можно рассматривать как модульное программирование нового уровня, когда вместо во многом случайного, механического объединения процедур и данных главным становится их смысловая связь. Объект рассматривается как логическая единица, которая содержит данные и правила (методы) их обработки. Объектно-ориентированный язык создает «программное окружение» в виде множества независимых объектов, каждый из которых отличается своими свойствами и способами взаимодействия с другими объектами. Программист задает совокупность операций, описывая структуру обмена сообщениями между объектами. Как правило, он «не заглядывает» внутрь объектов, но при необходимости может изменять элементы внутри объектов или формировать новые.

ООП основано на трех важнейших принципах (инкапсуляция, наследование, полиморфизм), придающих объектам новые свойства. *Инкапсуляция* — объединение в единое целое данных и алгоритмов их обработки. Данные здесь — поля объекта, а алгоритмы — объектные методы. *Наследование* — свойство объектов порождать своих потомков. Объект-потомок автоматически наследует все поля и методы, может дополнять объекты новыми полями, заменять и

дополнять методы. *Полиморфизм* — свойство родственных объектов решать схожие по смыслу проблемы разными способами.

Идея использования программных объектов исследовалась в течение ряда лет разными учеными. Одним из первых языков этого типа считают Simula-67. А в 1972 г. появился язык Smoltalk, разработанный Аланом Кеем, утвердивший статус ООП.

На современном этапе развиваются инструментальные среды и системы визуального программирования для создания программ на языках высокого уровня: (Turbo Pascal, Delphi, Visual Basic, C++Builder и др.).

Идея переложить на ЭВМ функции составителей алгоритмов и программистов дала новые возможности развитию сферы искусственного интеллекта, которая должна была создавать методы автоматического решения интеллектуальных задач. Формализация знаний, которые есть у профессионалов в разных областях, накопление их в базах знаний, реализованных на ЭВМ, стали основанием для создания экспертных систем. На основе баз знаний работают и ЭВМ V поколения, и интеллектуальные роботы, и экспертные системы. Эти системы могут не только найти решение той или иной задачи, но и объяснить, как оно получено. Появилась возможность манипулировать знаниями, иметь знания о знаниях — метазнания. Знания, хранящиеся в системе, стали объектом ее собственных исследований.

Независимость языков высокого уровня от ЭВМ вовлекла в сферу алгоритмизации задач специалистов различных отраслей знаний, позволила использовать многочисленные стандартные типовые программы, а программистам — устранять дублирование в написании программ для различных типов ЭВМ и значительно повысить производительность труда.

В конце 1980-х гг. в Японии и США появились проекты ЭВМ V поколения, реализованные в конце 1990-х гг. Прогресс в программировании связан с прогрессом в архитектуре вычислительных систем, отходом от фон-неймановской концепции, с достижениями в области искусственного интеллекта. Революционные изменения в элементной базе ЭВМ связываются с исследованиями по биоэлектронике.

На современном этапе программирование включает комплекс вопросов, связанных с написанием спецификаций (условий задач), проектированием, кодированием, тестированием и функционированием программ для ЭВМ. Современное ПО для ЭВМ имеет сложную структуру и включает, как правило, ОС, трансляторы с различных языков, текстовые программы контроля и диагностики, набор обслуживающих программ. Например, японские ученые для проек-

тирования систем ПО разрабатывают идею «кольцевой структуры» шести уровней: 1-й (внутренний) — программы для аппаратуры; 2-й — ядро ОС; 3-й — программы сопряжения; 4-й — часть ОС, ориентированная на пользователя; 5-й — системы программирования; 6-й (внешний) — программы пользователя.

Согласно этим проектам научных исследований планируется упростить процесс создания программных средств путем автоматизации синтеза по спецификациям исходных требований на естественных языках. В последнее время в Японии удалось создать робота-переводчика, переводящего английскую речь на японский язык и наоборот, осуществляя это голосом человека. Во всех развитых странах работают над комплексами программ для создания роботов для многих сфер человеческой деятельности.

Широкое применение структурных и объектно-ориентированных методов программирования с использованием графических моделей объединялось отсутствием инструментальных средств. Это породило потребность в программно-технологических средствах специального класса — CASE (Computer Aided Software Engineering), реализующих технологию создания и сопровождения ПО различных систем. Предпосылки для появления CASE-технологий возникли к концу 1980-х гг. Первоначально термин «CASE» применялся только к вопросам автоматизации разработки ПО, теперь программная инженерия имеет более широкое значение для разработки систем в целом. В CASE-технологии входит разработка и внедрение языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описания системных требований.

В начале XX в. с созданием пишущей механической машинки появилась возможность общедоступного создания печатного текста, хотя внесение изменений в такой текст (исправление ошибок) было достаточно трудоемкой работой. Затем появились электрические пишущие машинки. С появлением персональных компьютеров подготовка печатного текста стала гораздо совершеннее. В последние два десятилетия прошлого века уже разрабатывается множество комплексов программ для обработки текстов, которые сначала получили название *текстовых редакторов*, а по мере расширения их функциональных возможностей — *текстовых процессоров*.

В начале этого столетия текстовые процессоры стали более совершенными. Наряду с более простыми (например Professional Write и др.) появились такие мощные, как MS WinWord (см. рис. 21), WordPerfect WordStar 2000 и др. Из отечественных широкое распространение получил текстовый процессор Лексикон.

С начала 1980-х гг. для подготовки и обработки числовой информации стали использоваться *табличные процессоры*. В 1979 г. Д. Брикклин предложил первую программу для работы с электронными таблицами VisiCalc. В 1981 г. была разработана система SuperCalc фирмы «Computer Associates», в 1982 г. — Multiplan фирмы «Microsoft», далее — пакет для IBM PC Lotus1-2-3 фирмы «Lotus Development», русифицированные пакеты АБАК, ДРАКОН и др. В 1985 г. появился табличный процессор Excel фирмы «Microsoft» первоначально для персонального компьютера Macintosh, а затем для совместимых с IBM PC. Этот процессор разрабатывался параллельно с ОС Windows, его версии вобрали в себя все черты графического интерфейса, вплоть до версий Excel 5.0 как приложения Windows 3.1, Excel 7.0 как приложения Windows 95 и т. д. В последние годы создано достаточно много систем подготовки табличных документов, т. е. электронных таблиц, табличных процессоров (например, Corel Quattro 6.0 фирмы «Corel Co», Lotus 5.0 фирмы «Lotus Development Co», Office Professional for Windows фирмы «Microsoft» и др.). Но наиболее широко используют электронные таблицы Excel.

Разработано большое количество стандартных реляционных систем управления базами данных — СУБД (например, MS Access, Paradox и др.), на основе которых строят реляционные базы данных в различных предметных областях.

Для многих организаций (особенно управленческих) разработаны так называемые офисные пакеты, в которых на основе единой ОС функционируют приложения, включающие в себя системы для работы с различными видами информации. Например, созданы пакеты приложений к ОС Windows (MS Office, WordPerfect Office фирмы «Corel», StarOffice фирмы «SunMicrosystems» и др.), которые включают программные средства для выполнения функций обработки всех видов информации. Например, MS Office включает совершенствующиеся год от года (в зависимости от последней версии ОС Windows) средства обработки текста (MS Word), графики (Photo Draw) и презентаций (PowerPoint), таблиц (Excel), баз данных (Access), электронной почты (Outlook), работы во Всемирной паутине (FrontPage), создания звуковых клипов (MS Sound Recorder).

Мощным толчком в развитии новых направлений в программировании послужило объединение компьютерных и телекоммуникационных технологий.

За рубежом в 1960-х гг. появились первые вычислительные сети, с которых началась техническая и технологическая революция, т. к. была предпринята попытка объединить технологию сбора, хране-

ния, передачи и обработки информации на ЭВМ с техникой связи. В Европе в те годы были созданы международные сети EIN и Евро-нет, затем появились национальные сети. В 1972 г. в Вене была создана сеть МИПСА, к которой присоединились в 1979 г. 17 стран Европы, СССР, США, Канада и Япония. В 1980-х гг. в нашей стране была создана система телеобработки статистической информации, обслуживающая государственные и республиканские органы статистики. С 1980-х гг. развивается программирование для локальных вычислительных сетей (ЛВС).

ЛВС — это коммуникационная система, которая поддерживает в пределах одного здания или некоторой ограниченной территории один или несколько высокоскоростных каналов передачи информации, предоставляемых абонентским системам для кратковременного пользования. К 1990 г. эксплуатировалось свыше 0,5 млн серверов и 5 млн рабочих станций, работающих под управлением сетевых ОС (например NetWare компании «Novell»).

Глобальные вычислительные сети — это сети, использующие информационные ресурсы ЛВС, расположенных на большом расстоянии друг от друга (передача осуществляется с помощью телефонной сети через модемы или по выделенным каналам). Наиболее популярной является сеть Интернет, представляющая собой общемировую совокупность сетей, связывающая между собой миллионы компьютеров.

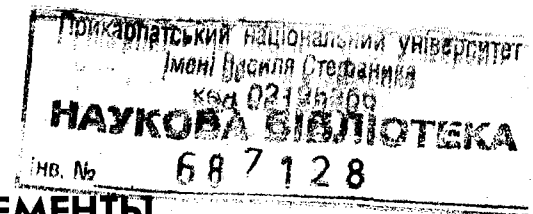
Сети позволили эффективно использовать аппаратные средства, программные средства и такие многопользовательские системы, как электронная почта, информационные системы на основе баз данных, телеконференции и др. Особой популярностью пользуется система WWW (World Wide Web) — Всемирная паутина, т. е. всемирная распределенная база гипертекстовых документов. Пользователи, используя для программирования язык гипертекстовой разметки HTML, создают свои сайты любой тематики и легко могут получать многообразную информацию, общаться с миллионами пользователей компьютеров. В будущем планируется массовое использование так называемых информационных роботов (Knowbot) — новых систем поиска и обработки информации в сети, в основе которых имеются уже элементы экспертных систем, позволяющих анализировать искомую информацию и готовить ее для выдачи в форме презентаций.

С Интернетом тесно связаны понятия «киберпространство» и «виртуальная реальность». *Киберпространством* называют совокупность всех систем компьютерных коммуникаций и потоков информации, циркулирующих в мировых сетях. *Виртуальная реальность* —

фантастический мир, создаваемый на экране компьютера, образы реального мира и процессов, в нем происходящих. С этими объектами и процессами можно работать как с реальными, проводить различные исследования, имитировать всевозможные ситуации, создавать прекрасные тренажеры для применения полученных навыков в реальности. Поле деятельности для программистов огромное, поэтому общество заинтересовано в высококвалифицированных специалистах этого профиля.

Глава 1

ОСНОВНЫЕ ЭЛЕМЕНТЫ ПРОГРАММИРОВАНИЯ



1.1. Управление компьютером с помощью программ. Система команд исполнителя. Алгоритмы. Программы. Машинные коды

Процессы управления в различных системах сводятся к реализации определенных алгоритмов. *Алгоритм* — одно из фундаментальных понятий программирования, включающий совокупность правил и указаний, сформулированных на некотором языке и направленных на достижение определенной цели (пример — рецепт, программа).

Слово «алгоритм» происходит от *algorithmic* — латинской формы написания имени великого математика IX в. Аль Хорезми, который сформулировал правила выполнения основных арифметических действий над многозначными числами.

С понятием «алгоритм» тесно связано понятие «*исполнитель*». Чтобы достичь цели, алгоритм должен быть кем-то или чем-то исполнен. Это может быть человек, механическое устройство, робот, компьютер и другие, способные понимать и выполнять команды алгоритма. Каждый исполнитель имеет свою систему команд — конечный перечень доступных пониманию указаний. Эта совокупность команд называется *системой команд исполнителя (СКИ)*.

Пример. Робот должен выполнить следующее задание: продвигаясь по цеху, взять деталь и установить ее.

Представим цех в виде клеток (рабочих полей), по которым должен продвигаться робот. Стрелки указывают направление его движения (рис. 1).

Исполнитель действует формально (не вникая в содержание, выполняет некоторые правила, инструкции), но получает требуемый результат. Следовательно, алгоритм формализует процесс ре-

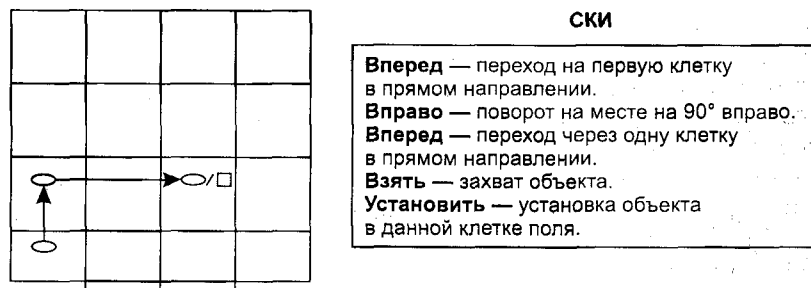


Рис. 1

шения задачи, позволяя механически использовать команды алгоритма в указанной последовательности. Итак, алгоритм — понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение поставленной цели.

Алгоритм должен быть *понятным*, *дискретным* (состоять из отдельных законченных команд), *результативным* (выполняться за конечное число шагов), *определенным* (каждая команда и правило алгоритма должны быть четкими, однозначными, не оставляющими места для произвола).

Виды и запись алгоритмов. Существуют алгоритмы работы с величинами — числовыми, символьными, логическими и алгоритмы работы «в обстановке» (например робот).

Они должны быть записаны на алгоритмическом языке, т. е. представлены в виде графического и (или) словесного описания, таблицы, последовательности формул, на учебном алгоритмическом языке, языке программирования. Иногда их записывают на псевдокодах (специальных языках).

Алгоритмический язык — это система обозначений и правил для единообразной и точной записи алгоритмов и их исполнения. Язык записи алгоритма должен быть понятным для человека.

Наиболее распространен специальный графический язык описания алгоритмов — структурные схемы. Схема в стандарте определена как «графическое представление определения, анализа или метода решения задачи, в котором используются символы для отображения операций, данных, потока, оборудования и т. д.» Существуют схемы данных, программ, работы системы, взаимодействия программ, ресурсов системы. Схемы обеспечивают наглядность, читаемость, отображение последовательности выполняемых процессов. Схема — это ориентированный граф, стрелками (или линиями) указывающий порядок исполнения команд алгоритма, а вершины (со-

бытия) такого графа представлены геометрическими фигурами, которые называются символами. Например, начало и конец записи алгоритма обозначаются овалом, данные (носитель которых не определен) — параллелограммом, процесс (обработка данных любого вида) — в виде прямоугольника, решение (или функция переключательного типа, например условие) — в виде ромба и т. д. (рис. 2). Стандарт на этот специальный графический язык для записи дан в ЕСПД (Единая система программной документации, стандарт — ГОСТ 19.701—90).

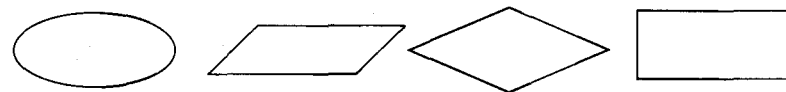


Рис. 2

В основе построения алгоритмических структур лежит теорема *структурного программирования*, включающая следующие основные принципы:

- всякий реальный алгоритм может быть построен с использованием трех базовых элементов: следования, ветвления и цикла;
- любая алгоритмическая структура, состоящая из базовых элементов, может быть представлена как единый процесс;
- алгоритм проектируется по нисходящей схеме;
- поэтапное уточнение или пошаговая детализация алгоритма.

В графической форме базовые вершины могут быть одного из трех типов:

- функциональная (один вход и один выход);
- предикатная (один вход и два выхода);
- объединяющая (два выхода и один вход, передающий управление от первого из двух выходов).

Из данных элементарных схем можно построить четыре схемы основных алгоритмических структур, имеющих особое значение для практики алгоритмизации.

Схема *а* на рис. 3 — самая простая структура алгоритма — это последовательность, если команды следуют одна за другой в естественном порядке. Такой алгоритм называется *линейным*. Здесь S1 и S2 — некоторые серии команд (S — функциональные вершины).

Схема *б* на рис. 3 — структура, в которой порядок следования команд определяется в зависимости от результатов проверки некоторых условий. Такой алгоритм называется *разветвляющимся*. В алго-

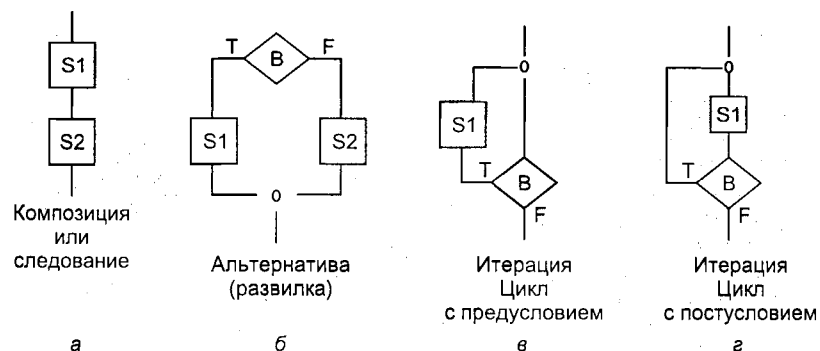


Рис. 3. Схемы основных алгоритмических структур: *а* — линейный алгоритм; *б* — разветвляющийся алгоритм; *в, г* — циклические алгоритмы с предусловием и постусловием соответственно

ритме *разветвляющейся структуры* действия записываются не подряд, а в зависимости от выполнения (невыполнения) некоторого условия. Здесь В — условие (предикатная вершина), в зависимости от истинности (Т) или ложности (F) которого управление передается по одной из двух ветвей. Объединяющая вершина обозначена — О.

Схемы *в, г* на рис. 3 — структуры, в которых получение результата обеспечивается многократным повторением одних и тех же операций. Это — *циклические алгоритмы с предусловием и постусловием* (о них подробнее см. 1.3).

Схему (блок-схему) алгоритма можно изобразить, например, вызвав через меню текстового процессора MS Word команды: Вид → Панели инструментов → Рисование с помощью *Автофигур* (группа Блок-схема), где представлены начертания всех стандартных фигур и их обозначения. Надписи в фигурах следует создать с помощью команд: Вставка → Надпись.

Типовая блок-схема алгоритма *линейной структуры* показана на рис. 4. Типовая блок-схема алгоритма *разветвляющейся структуры* представлена на рис. 5.

В схеме разветвления алгоритма операцию проверки условия выполняет логический блок, изображенный ромбом, внутри которого указывается проверяемое условие (отношение), и имеется два выхода: ДА и НЕТ. Если условие истинно, то выходим на ДА, если ложно — то НЕТ. Типовая блок-схема алгоритма *циклической структуры* показана на рис. 6.

Примеры блок-схемы линейного алгоритма для решения конкретных задач см. на рис. 8, разветвляющегося — на рис. 7 и 9, циклического — на рис. 10.



Рис. 4

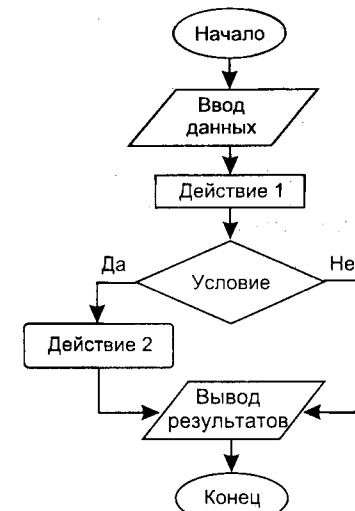


Рис. 5

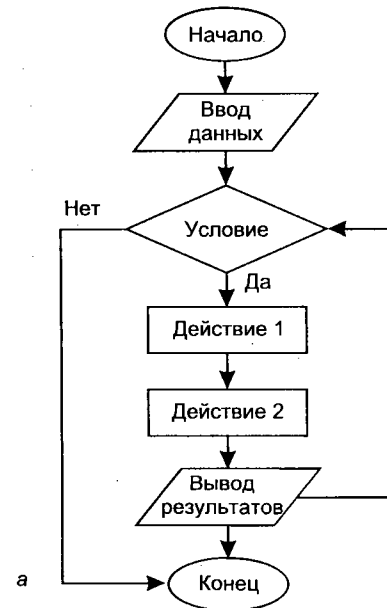


Рис. 6. Блок-схемы алгоритмов циклической структуры: *а* — цикл с предусловием; *б* — цикл с постусловием

Примеры блок-схемы алгоритма циклической структуры с пред-условием и с постусловием для решения конкретной задачи представлены соответственно на рис. 14 и 15.

Для составления алгоритма следует:

- осмыслить условия (требования) задачи, т. е. выяснить, что является исходными данными, какие из них допускаются в задаче, что является результатом;
- составить строгую формулировку задачи по следующей форме: задача, аргументы, ограничения, результаты;
- записать описательную часть алгоритма и наметить план решения, т. е. процедурную часть, составив ее в произвольной словесной форме;
- переписать алгоритм, используя допустимые для предполагаемого исполнителя команды, оставляя неясные места;
- многократно переписывать алгоритм, каждый раз уделяя внимание вопросам, оставшимся от предыдущего шага, более мелким деталям, пока не получится текст, по которому можно писать программу для исполнителя (не обязательно компьютера).

От алгоритма к программе. Чтобы алгоритм стал понятен ЭВМ, его следует *закодировать* — перевести на строго формализованный язык, т. е. язык программирования. Кодирование — процесс достаточно механический, но требует твердого знания команд и синтаксиса языка программирования.

Алгоритм, записанный на языке программирования, называется *программой для ЭВМ*. Языки эти формальные, специально созданные для общения человека с компьютером. Каждый язык программирования имеет алфавит, словарный запас, грамматику, синтаксис и семантику. *Алфавит* — фиксированный набор символов, допускаемых для составления текста программы на этом языке. *Синтаксис* — система правил, определяющих допустимые конструкции языка программирования из букв алфавита. *Семантика* — система правил однозначного толкования отдельных языковых конструкций, позволяющих воспроизвести выполнение процесса обработки данных. Если программа написана в машинных кодах, то она может сразу исполняться ЭВМ. Но писать такие программы сложно, они очень громоздки и плохо воспринимаются человеком. Если формальный язык, на котором написана программа, хорошо понятен человеку, но не может сразу быть воспринят компьютером, то создают специальные программы (трансляторы: компиляторы или интерпретаторы), осуществляющие автоматический перевод программ с языка программирования в машинные коды.

Предположим, что *исполнителем* алгоритма будет ЭВМ, т. е. электронный автомат. Компьютер, к сожалению, не воспринимает команды на естественном языке. Для управления компьютером необходимо освоить специальный командный язык. Внутри себя компьютер составляющими его микросистемными устройствами командует на внутреннем машинном языке, удобном для микросистемной электроники. Людям трудно общаться на этом языке, его понимают только специалисты. Для общения между компьютером и человеком, точнее, для приказов компьютеру что-то делать, изобретены специальные языки, названные языками программирования. На этих языках составляются программы — совокупности команд для управления компьютером. До 1980-х гг. компьютеры использовались для математических и технических расчетов и действий с экономической информацией. С этой целью были разработаны проблемно-ориентированные языки программирования (Fortran, Algol, Cobol). Дальнейшее развитие языков делало их более универсальными, позволяющими составлять программы для любых информационных действий, управления печатью, рисования чертежей, воспроизведения звуков и т. д. Сейчас квалифицированные пользователи используют 2—3 командных языка: для профессионалов — C, для обучения — Pascal, Basic.

Программа — упорядоченный список команд (инструкций, включающих операторы и параметры) на языке программирования. Такая программа называется *исходным текстом* или *исходным кодом*. Для реализации программы она должна быть откомпилирована, в результате чего образуется *объектный код*, записанный в машинных кодах. Для подключения к программе необходимых стандартных процедур и функций используют программу *редактор связей*, которая выполняет эту работу, извлекая из библиотек стандартных подпрограмм необходимые и вставляя их в объектный код. Полученная программа называется *исполняемым кодом* и является уже *рабочей программой*, которую можно запустить на исполнение. Для записи исходного текста программы прежде всего необходимы три простые команды: присваивание, ввод, вывод. Команда *присваивания* служит для изменения состояния объектов алгоритма и обозначается, например в языке Pascal, символом := (например, X:=1, A:=B). Команды *ввода-вывода* необходимы для связи ЭВМ с внешним миром. Эти команды относятся к разряду основных и реализованных в любой машине. Основной единицей программы, выполняющей определенные действия над данными, является *оператор*. Операторы бывают управляющие и обрабатывающие.

В программах для ЭВМ используются описания объектов следующих типов: целые, вещественные, литерные (символьные) и др. С каждым типом связан свой способ представления объекта в памяти ЭВМ.

Основная структура программы на языке Паскаль имеет следующие разделы: заголовок, описание данных, начало программы, операторы, конец программы.

Здесь приведен пример записи алгоритма на учебном алгоритмическом языке и программы на языке Паскаль для решения конкретной задачи.

Пример. Найти периметр и площадь круга радиуса R .

Аргументы:

R — радиус круга, вещественное число.

Результаты:

L — периметр круга, вещественное число;

S — площадь круга, вещественное число.

Алг Парам (вещ R , L , S)

Арг R

Рез L , S

Нач

Ввод (R)

Определить периметр $L := 2\pi R$

Определить площадь $S := \pi R^2$

Вывод (L, S)

Кон

Program Prim-1 (Input, Output);

const

$P_i = 3.1416$

var

R, L, S : real;

begin

readln (R); {ввод R }

$L := 2 * P_i * R$; {вычисление L }

$S := P_i * \text{sqr} (R)$; {вычисление S }

writeln (L, S); {вывод L и S }

end.

Программы, записанные на любом языке программирования, сначала с помощью трансляторов переводят в *машинный код*. А с помощью программ-отладчиков, позволяющих находить ошибки в программе, можно посмотреть во время работы ее машинные коды. Машинный код записывается в шестнадцатеричной системе счисления, двузначными числами или обозначениями (например, 33, 41,

45, C0, F6 и т. д.), каждому из которых отведен байт, находящийся в своей ячейке памяти. Таким образом, программа в машинном коде — это набор байтов, которые процессор понимает и их различает: команды, числа, символы, адреса.

Более понятным программистам, чем машинный код, является специальный код — *код ассемблера*. Он записывается в виде мнемоник. Каждая команда — мнемоника (сокращенные слова английского языка). Например, машинный код, представленный числом 93 в виде мнемоники, записывается как EXCHG BX, AX и означает: обменять (EXCHG) содержимое регистров BX и AX. Иногда системные программы пишут на ассемблере, а потом переводят в машинный код — ассемблирование или наоборот, чтобы легче было читать программу, выполняют дисассемблирование.

Программы пишут специалисты — программисты, т. е. люди со специальной подготовкой, имеющие научные знания и владеющие определенным инструментарием в области создания программ. В начале программисты работали только с процессором, который понимает определенный (и каждый тип процессора свой) числовой машинный код. Каждый тип процессора имеет свою систему команд, из которых и создаются компьютерные программы. Но машинные коды сложны для восприятия человеком, поэтому постепенно стали развиваться языки, более приближенные к естественным. Программисты стали писать программы (*программировать*) в удобной форме, на понятных им языках программирования. А написанные специальные программы, которые могут прочитать то, что написал программист и перевести в машинные коды, стали инструментом программирования.

Постепенно программы накапливались, особенно на процессы, выполняемые наиболее часто (например ввод данных с клавиатуры и вывод информации на экран и т. д.). Такие небольшие программы (процедуры) называют *стандартными*. В настоящее время программист при создании большой программы обращается к библиотекам стандартных программ, извлекает необходимые программы-процедуры и включает их в свою программу. Это значительно облегчает процесс программирования.

С самого начала следует выработать хороший стиль программирования и оформления программ: при создании давать им содержательные имена (идентификаторы), использовать ступенчатую форму записи, каждый оператор размещать на отдельной строке, давать ясные комментарии, размещать текст программы только в рамках экрана и т. п.

Контрольные вопросы

1. Что такое алгоритм?
2. Что такое система команд исполнителя?
3. Что такое графический язык описания алгоритмов?
4. Что такое программа для ЭВМ?
5. Как от алгоритма перейти к созданию программы для компьютера?

1.2. Общие понятия о составлении программы и этапы ее разработки. Разделение программы на части

Любая задача начинается со словесного описания, которое называется условием задачи. Далее условие следует формализовать, т. е. так записать спецификацию, чтобы ее можно было решить на компьютере. Формализацией условия начинается этап, который называется математической формулировкой задачи. Выводятся формулы и выбираются методы решения задачи, т. е. строится ее математическая модель. Затем в виде алгоритма строится последовательность стандартных действий, выполнение которых даст искомый результат. На выбранном языке программирования проектируется и пишется программа для компьютера, выполняется ее отладка и тестирование на компьютере, а также необходимые расчеты.

Как начать разработку программы? Как правило, в любой программе компьютер должен выполнять ввод, обработку, хранение и вывод информации. Составление программы осуществляют набором констант, переменных, операторов и т. д., как правило, указывая их в определенной последовательности. Часто программа состоит из основной (главной) части и так называемых подпрограмм.

Изучение программирования начнем с помощью команд языка Паскаль (Pascal) в среде Турбо Паскаль (Turbo Pascal).

Задача. Вывести на экран текст: Сказка Репка.

```
Program Repka_1;
begin
  writeln ('Сказка Репка');
end.
```

В этой программе имеется только главная часть, которая находится между служебными словами *begin* (начало) и *end* (конец). Предложения языка программирования, задающие действия компь-

ютера, называются *операторами* (т. е. заданиями операции над информацией). В данном примере оператор *writeln* — оператор вывода.

Операторы вывода: *write (b : m)*, где *b* — имя выводимой переменной, *m* — число позиций, константа или выражение целого типа; *write(b;m;n)*, где *n* — дробная часть значения выводимой переменной.

Примеры: *x:=0.25*, *x:=x+0.25*; *t:=-10.6*, *a:= t*;

По формальным правилам языка, то, что надо вывести на экран, помещают в скобки (), а сам текст берется в апострофы. Если задана команда *writeln*, то экран настраивается на вывод в специальной строке, если *write* — то вывод в этой же строке экрана. В конце программы ставится точка.

Правила строгие, если их нарушить, система программирования выдаст сообщение об ошибке.

Усложним программу.

Задача. Вывести на экран 3 строки текста:

1) Сказка Репка; 2) Посадил дед Репку; 3) Выросла Репка большая-пребольшая.

```
Program Repka_2;
begin
  writeln ('Сказка Репка');
  writeln ('Посадил дед Репку');
  writeln ('Выросла Репка большая-пребольшая');
end.
```

Символ ; (точка с запятой) используется для разделения предложений языка (операторов).

Реальные программы содержат не только вывод результатов, но и ввод исходных данных для их компьютерной обработки, а также приказы на нее. В Паскале *оператор ввода* с клавиатуры — *read*, за которым в () указываются переменные через запятую. Завершение ввода — нажатие клавиши Enter. *Оператор присваивания* служит для вычисления значения выражения и присваивания его переменной, расположенной слева от символа :=. Общий вид записи оператора присваивания: *a:= b*, где *a* — имя переменной, *b* — выражение.

Задача. Ввести по запросу компьютера два числа, сложить их и вывести сумму на экран.

```
Program Dva;
var {служебное слово для описания раздела переменных}
  x,y,sum: integer; {тип переменных — целые}
```

```

begin
  writeln ('Наберите на клавиатуре два числа с пробелами
           между ними');
  read (x, y);      {ввод переменных}
  sum:= x+y;        {вычисление суммы}
  writeln ('Сумма=', sum);
end.

```

Собственно вычисление суммы в программе выполняет *оператор присваивания*. Он вычисляет выражение в правой части оператора и заносит результат в переменную в левой его части. Далее с помощью оператора `writeln` это значение выводится на экран с текстовым пояснением.

В фигурные скобки { } даются *комментарии*, которые предназначены для человека и не воспринимаются компьютером.

Имя переменной обязательно начинается с латинской буквы или символа подчеркивания, а далее могут быть цифры, буквы, подчеркивания. Например: `Репка_2` — одно имя.

Тип переменных определяет, какое значение может принимать эта переменная и какие операции над ней возможны. Наиболее простые типы: целые числа (`integer`), числа с дробной частью — вещественный (`real`), буквы, символы — символьный (`char`).

Простейшее применение программ на языках программирования — это расчеты по математическим формулам. В начале программы описывается переменная, затем приказ ввести конкретные значения для переменной формулы и далее записать эти формулы в виде, требуемом строгими правилами языка программирования. Основная часть математических формул, математические выражения — специальная форма записи математических действий над константами и переменными. Для записи математических формул на языке программирования используются следующие обозначения:

- * (звездочка) — знак умножения;
- / — знак деления и дополнительное применение скобок $(x+y)/(x-y)$;
- стандартные математические функции требуют обязательного указания аргументов в скобках — $\sin(x)$;
- корень квадратный записывается через функцию `sqrt`;
- переменные с индексами будут рассматриваться далее (см. информацию о массивах);
- используют некоторые функции, например: `Round` — для округления дробного значения до ближайшего целого, `Trunc` — для отбрасывания дробной части (`truncate` — усечь, обрезать) и др.

Например, в следующей программе наглядно представлена задача с расчетами по математическим формулам.

Задача. Вычислить значения корней квадратного уравнения: $x^2 + px + q = 0$.

```

Program Korni_1;
var
  p, q, d, x1, x2, : real;
begin
  write ('Введите коэффициенты квадратного уравнения');
  read (k, p, q);
  d:=p*p-4*q;
  if d>= 0 then
    begin
      x1:= -p/2 + sqrt (d);
      x2:= -p/2 - sqrt (d);
      writeln ('X1=', x1:5:2, ' X2=', x2:5:2);
    end;
  else writeln ('уравнение не имеет вещественных решений');
end.

```

Как было отмечено выше, *разработка программы* проходит следующие *этапы*.

1. Анализ требований, предъявляемых к ПО.
2. Определение спецификации.
3. Алгоритмизация (проектирование).
4. Кодирование (перевод алгоритма на язык программирования).
5. Отладка и тестирование.
6. Эксплуатация и сопровождение.

Рассмотрим этапы разработки программы для задачи, аналогичной предыдущей.

1. *Анализ требований, предъявляемых к ПО.* Составить алгоритм нахождения вещественных корней квадратного уравнения $kx^2 + px + q = 0$. Это уравнение имеет решение в действительных числах $x_{1,2} = (-p \pm \sqrt{d})/2k$, если его дискриминант $d = (p^2 - 4kq) > 0$. При $d=0$ корни этого уравнения определяются по формуле $x_1 = x_2 = -p/2 \cdot k$.

Причем при выполнении условия требуется вычислить оба корня.

2. *Определение спецификации.* Запись квадратного уравнения на алгоритмическом языке: $kx^2 + px + q = 0$.

Чтобы строго записать условия данной задачи (спецификацию алгоритма) надо знать следующее:

- исходные данные — это коэффициенты k, p, q ;

- уравнения будут только в случае, если $k \neq 0$;
- вещественные корни будут найдены только при условии $p^2 - 4kq > 0$;
- корни два.

Аргументы: k, p, q — коэффициенты уравнения, вещественные числа.

Ограничения: $k \neq 0$ — условие существования квадратного уравнения. $p^2 - 4kq \geq 0$ — условие существования квадратного уравнения.

Результаты: x_1, x_2 — корни уравнения, вещественные числа.

t — сообщение исполнителя, литерная строка.

t = 'квадратное уравнение не существует' или 'вещественных корней нет', или 'вещественные корни не найдены'.

3. *Алгоритмизация.* Описательная (декларативная) запись алгоритма и содержательная (процедурная) части алгоритма.

Алг веш кор (веш k, p, q, x_1, x_2 , стр t)

Арг. k, p, q

Рез x_1, x_2, t

Нач

Ввести аргументы

Решить квадратное уравнение {процедурная часть}

Вывести результаты

Кон

В виде схемы алгоритм будет иметь следующий вид (рис. 7).

4. *Кодирование (перевод алгоритма на язык программирования).*

Program Korn1_2;

var

k, p, q, d, x_1, x_2 ; : real;

begin

write ('Введите коэффициенты квадратного уравнения');

read (k, p, q);

$d := p^2 - 4 * k * q$;

if $d > 0$ then

begin

$x_1 := (-p + \sqrt{d}) / (2 * k)$;

$x_2 := (-p - \sqrt{d}) / (2 * k)$;

writeln (' $x_1 =$ ', x_1 :5:2, ' $x_2 =$ ', x_2 :5:2);

end

else writeln ('Уравнение не имеет вещественных решений')

end.

И далее выполняются пункты 5 и 6 (см. тему 1.4).

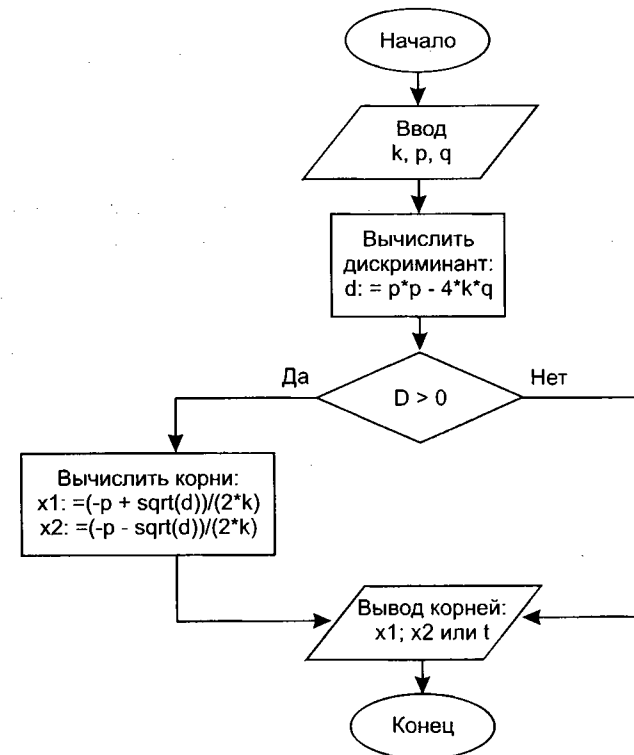


Рис. 7

Разделение программы на части. Сложные программы почти невозможно создать без их разделения на относительно независимые части: *основная* часть программы и *подпрограммы* — именованные группы программных действий, которые могут быть вызваны из других мест программы. Подпрограммам дают уникальные имена. Они должны быть описаны один раз и многократно использованы.

Имена переменных, описываемых непосредственно после заголовка и до тела подпрограммы, называются *локальными*. Они допустимы для использования только в той подпрограмме, где они описаны. *Имена переменных,* описанных перед всеми подпрограммами, называются *глобальными*. Они должны быть доступны для использования как в подпрограммах, так и в основной части программы. Принято соглашение: внутри подпрограммы действуют локальные

переменные, снаружи — глобальные. Глобальные имена следует применять очень осторожно.

Подпрограммы условно разделяют на *подпрограммы-функции* и *подпрограммы-процедуры*. Но это относится не ко всем языкам программирования.

Подпрограммы-функции. Подпрограмма-функция обязательно возвращает результат, который затем используется в программе. Тип результата должен быть задан. Для обозначения подпрограммы используют служебное слово *function*.

Задание компьютеру действий (т. е. функции предназначения) в месте, отличном от места их использования, называют описанием подпрограммы.

Общий вид описания подпрограммы-функции на языке Pascal:

```
function имя_функции (список_аргументов): тип результата
  описание внутренних переменных подпрограммы
begin
  тело подпрограммы (операторы, задающие действия
    с использованием формальных аргументов).
end;
```

Подпрограмма по своему строению напоминают простую программу.

Приведем несколько примеров таких программ.

Задача. Определить минимальное из трех заданных чисел.

Вначале пишем подпрограмму—функцию определения минимального из двух заданных чисел. Описание подпрограммы вводит задание действий над абстрактными аргументами, которыми в примере являются переменные *x* и *y*, называемые формальными аргументами, или формальными параметрами. Они описываются не в основной части программы — после служебного слова *var*, а непосредственно в заголовке описания подпрограммы.

```
Program Iz_3_min;
function min (x:integer; y: integer):integer;    {описание}
                                              {подпрограммы min}
begin
  if x<y then
    min:=x
  else min:=y;
end;      {конец описание подпрограммы-функции}
          {Затем используем ее в программе}
```

```
var
a, b, c, min3 :integer;
begin
  write ('Введите 3 числа');
  read (a, b, c);
  writeln ('Минимальное из первых двух есть', min (a,b));
  writeln ('Минимальное из первого и третьего есть', min(a, c));
  min3:=min(a, min(b,c));
  writeln('Минимальное из трех есть', min3);
end.
```

Задача. Вычислить площадь треугольника.

```
Program Plotchad_1; {с подпрограммой-функцией}
function streug (a,b,c: real):real;
var
p: real;    {полупериметр}
begin
  p:=(a+b+c)/2;
  streug:=sqrt(p*(p-a)*(p-b)*(p-c))
end;
var
x1, x2, x3: integer;
begin
  writeln ("Площадь Δ со сторонами 10 =; streug (10,10,10):6 :3);
  writeln ("Введите длины сторон треугольника");
  read (x1, x2, x3);
  writeln ('Площадь Δ с этими
    сторонами =', streug (x1, x2, x3):7:2);
end.
```

Подпрограммы-процедуры. Эти подпрограммы не возвращают ни одного значения или возвращают более одного значения. Для обозначения таких подпрограмм используют служебное слово *procedure*, а задание типа результата отсутствует.

Общий вид описания подпрограммы-процедуры на языке Pascal:

```
procedure имя_процедуры (список_аргументов);
  описание внутренних переменных программы
begin
  тело подпрограммы
end;
```

Имя процедуры не может использоваться внутри процедуры как псевдопеременная.

Рассмотрим построение программы по условию предыдущей задачи с построением подпрограмм-процедур. Для существования

треугольника необходимо, чтобы сумма любых его сторон была не меньше третьей. Таким образом, подпрограмма должна возвращать два значения: логическое значение — существует ли треугольник и само значение площади треугольника.

Задача. Вычисление площади треугольника.

```

Program Plotchad_2; {с подпрограммой-процедурой}
var
x1, x2, x3: integer; {Возвращаемые аргументы}
procedure ptreug (a, b, c: real; var streug: real; var est: boolean);
{Формальные аргументы}
var
p: real;
begin
if (a+b<c) or (a+c<b) or (b+c<a) then
est:=false
else
begin
est:= true;
p:=(a+b+c)/2;
streug:=sqrt((p*(p-a)*(p-b)*(p-c))
end
end;
var {здесь задаются переменные только для главной части
программы}
s1, s2: real;
is: boolean;
begin
ptreug (10, 10, 10, s1, is); {процедуры названы именем ptreug}
writeln ('Площадь Δ со сторонами, равными 10, равна', s1:6:3);
writeln ('Введите длины сторон треугольника');
read (x1, x2, x3);
ptreug (x1, x2, x3, s2, is);
if is:= true then
writeln ('Площадь треугольника с этими сторонами равна', s2:7:2);
else writeln ('Треугольника с такими сторонами не существует')
end.

```

Контрольные вопросы

1. Как начать разработку программы?
2. Какие вы знаете операторы?
3. Что такое имя и тип переменной?
4. Что такое подпрограмма-процедура?
5. Что такое подпрограмма-функция?

1.3. Виды и этапы создания программных продуктов

Программы разрабатываются в соответствии с типами алгоритмов: линейные, разветвляющиеся, циклические, вложенных циклов. Сложные программы могут включать все типы структур.

Программы *линейной* структуры строятся как последовательность выполняемых друг за другом операторов. Линейная программа не содержит каких-либо условий. Как правило, в таких программах требуются операторы: ввода информации, присваивания, вывода результатов вычислений.

Задача. Даны длины двух катетов (A и B) прямоугольного треугольника. Определить периметр этого треугольника ($P = A + B + C$), где гипотенуза $C = \sqrt{A^2 + B^2}$ (рис. 8).

```

Program Perimetr;
var
A, B, C: real;
begin
writeln ('Введите значения катетов A и B');
read (A, B);
C:=(A^2+B^2)^0,5;
P:=C+A+B;
writeln (' Гипотенуза C=', C:4:2);
writeln ('Периметр P=', P:5:2);
end.

```

Далее запускаем программу на выполнение с помощью системной команды RUN. Исправляем ошибки, т. е. выполняем отладку программы, т. е. результат решения задачи:

'C='; C, 'P='; P

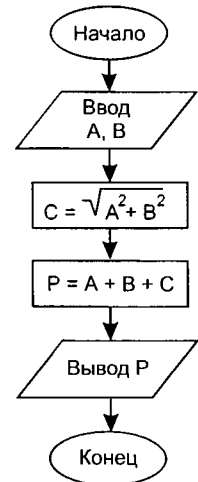


Рис. 8

Программы *разветвляющейся структуры* предусматривают выбор одного из нескольких последовательностей операторов (ветвей) в зависимости от некоторых условий.

Для организации разветвлений используются операторы: перехода, условные, выбора.

1) *Оператор безусловного перехода*: GO TO n , где n — метка.

Пример: GO TO 5 управление передается оператору, помеченному 5, и далее выполняются операторы за операторами перехода. (Сейчас этот оператор редко используется.)

2) Условный оператор IF b THEN a или IF b THEN $a1$ ELSE $a2$, где a , $a1$, $a2$ — операторы (могут быть простые и составные, если a имеет значение TRUE, то вычисляется a , иначе a не выполняется, а выполняется оператор, стоящий за условным).

В задачах часто приходится иметь дело с выбором дальнейших действий.

Условия выбора задаются на языке программирования. Берутся полные смысловые аналоги английского языка для союзов русского языка *если, то, иначе* — IF, THEN, ELSE. Например:

```
Program Gradus;
var
  temper: integer;
begin
  write ('Сколько градусов на улице?');
  read (temper);
  if temper > 20 then
    writeln ('Оденьтесь полегче, будет жарко!');
  else writeln ('Наденьте что-нибудь из верхней одежды!');
end;
end.
```

Как отмечалось ранее, создание программ для решения задач на ЭВМ проходит этапы, в которых определяются: постановка задачи, метод решения, сценарий работы с ЭВМ, алгоритм, перевод алгоритма в программу (кодирование), ввод и тестирование программы, анализ полученных результатов.

Рассмотрим примеры создания программ в соответствии с перечисленным.

Задача. Найти большее из двух чисел.

Постановка задачи.

Дано: a — целое положительное число; b — целое положительное число.

Найти: большее из двух чисел.

Метод решения.

Программирование следует начинать со сценария — определения числа вариантов решения задачи, типа переменных, их разрядности, порядка ввода, вывода результата.

Сценарий.

1) Наметить вариант сравнения двух целых положительных чисел, присваивая большему из чисел значение m .

2) Записать алгоритм сравнения двух целых положительных чисел на естественном алгоритмическом языке и в виде блок-схемы (рис. 9).

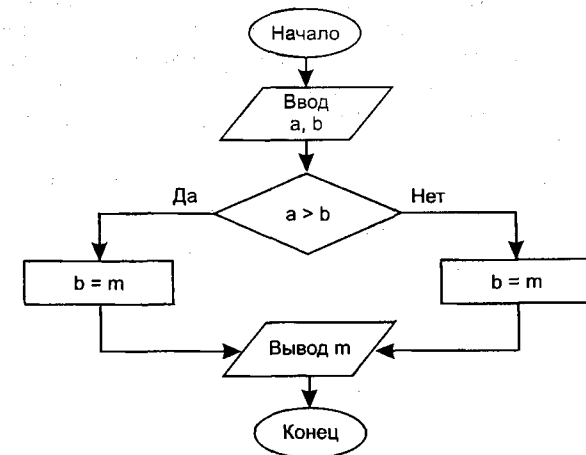


Рис. 9. Блок-схема

Алгоритм.

АЛГ Больше из двух чисел (арг цел a, b рез цел c)

НАЧ

вывод ("два числа")

ввод (a, b)

если $a > b$

то $m := a$

иначе $m := b$)

все

вывод ("больше", m)

КОН

Программа нахождения большего из двух чисел.

Program Bol_2;

var

a, b, m : real;

begin

writeln ('Введите два числа: a и b ');

readln (a, b);

if $a > b$ THEN

$m := a$

else

$m := b$

end;

writeln (' Больше =', m :4:2);

end.

Запускаем программу на выполнение с помощью системной команды RUN.

Исправляем ошибки, т. е. выполняем отладку программы.

Результат выполнения программы, т. е. распечатка на экране: 'Большее ='; m.

3) *Оператор выбора CASE* — обеспечивает организацию разветвлений путем выбора одного из нескольких операторов. Оператор выбора имеет список с элементами (операторами), перед которыми записываются одна или несколько констант, отделяемых двоеточием.

Пример.

```
Program Sckola
var
K: integer;
begin
  writeln ('Введите номера классов');
  readln (K);
  case K of
    1,2,3,4: writeln ('Начальные классы');
    5,6,7,8: writeln ('Средние классы');
    9,10,11: writeln ('Старшие классы')
  end;
end.
```

После запуска программы на выполнение и отладки получим результат, т. е. распечатку на экране дисплея:

Введите номера классов

Вводим цифры, например от 1 до 3. Ответ на экране: *Начальные классы*

Программа циклической структуры позволяет многократно вычислять группу операторов при одновременном изменении одного или нескольких параметров. Встречаются циклы с известным и неизвестным числом повторений.

Могут быть использованы условный оператор и оператор перехода, но в языке Pascal есть специальные операторы цикла: с *параметром FOR*, с *предусловием* — WHILE, с *постусловием* — REPEAT.

Оператор цикла FOR служит для организации цикла с известным числом повторений.

Записывается:

FOR $i := m_1$, TO m_2 DO S;

i — параметр цикла, величина целого типа; шаг цикла = 1; m_1 и m_2 — начальные и конечные значения параметров цикла; S — тело цикла.

Если ключевое слово TO заменить на DOWN TO, то шаг цикла = -1.

Оператор цикла WHILE <параметр> или <выражение> DO S позволяет организовать цикл с неизвестным числом повторений, т. к. оно зависит от вычислений в операторе. S — тело цикла (один или несколько операторов).

Если логическое выражение TRUE, то выполняются операторы тела цикла.

Значения переменных, входящих в условие, должны изменяться в теле цикла, иначе оператор никогда не завершится.

Оператор цикла REPEAT ... UNTIL (как и while), но условие проверяется в конце каждой итерации.

Далее приведены примеры программ с разными операторами цикла для одной и той же задачи.

Задача. Вывести десять значений целого типа, начиная с единицы, в возрастающем порядке.

1-й вариант.

```
PROGRAM NUM 10 (OUTPUT);
CONST KN=10;
VAR N: INTEGER;
BEGIN
  FOR N:=1 TO KN DO;
    WRITE (N: 6)
  END;
```

2-й вариант (с алгоритмом) представлен на рис. 10.

```
PROGRAM NUM 10 (OUTPUT);
CONST KN=10;
VAR N: INTEGER;
BEGIN
  N:=0;
  WHILE N<KN DO
  BEGIN
    N:=N+1;
    WRITE (N);
  END;
END.
```

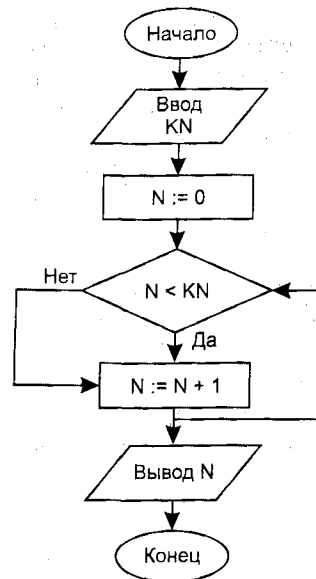


Рис. 10

3-й вариант.

```

PROGRAM 10 (OUTPUT);
CONST KN=10;
VAR N: INTEGER;
BEGIN
  N:=0;
  REPEAT
    N:=N+1;
    WRITE (N);
  UNTIL N>=KN
END;
END.
  
```

В соответствии со стандартами ГОСТ 19.101—77, а также 19.781—90 программа может разрабатываться и применяться самостоятельно и (или) в составе других программ.

Программы подразделяют на следующие виды:

- *компонент* — программа, рассматриваемая как единое целое, выполняющая законченную функцию и применяемая самостоятельно или в составе комплекса;
- *комплекс* — программа, состоящая из двух и более компонентов и (или) комплексов, выполняющих взаимосвязанные

функции, применяемая самостоятельно или в составе другого комплекса.

Для реализации и передачи программы на носителях данных, а также для изготовления программных изделий (продуктов) разрабатывается документация на программу. Программные документы должны содержать сведения, необходимые для разработки, изготовления, сопровождения и эксплуатации программ. Существуют следующие виды программных документов и их содержание:

- спецификация — состав программы и документации на нее;
- ведомость держателей подлинников — перечень предприятий, на которых хранят подлинники программных документов;
- текст программы — запись программы с необходимыми комментариями;
- описание программы — сведения о логической структуре и функционировании программы;
- программа и методика испытаний — требования, подлежащие проверке.

Более подробно о программной документации см. в теме 6.2.

Разработка программ и программной документации независимо от их назначения и области применения должна пройти следующие стадии и этапы работ.

1. *Техническое задание*: обоснование необходимости разработки программы, научно-исследовательские работы, разработка и утверждение технического задания.

На первом этапе должна быть четко сформулирована постановка задачи, собраны исходные материалы, выбраны и обоснованы критерии эффективности и качества разрабатываемой программы. На втором этапе следует обосновать необходимость проведения научно-исследовательских работ, определить структуру входных и выходных данных, предварительно выбрать методы решения задач, обосновать целесообразность применения ранее разработанных программ, определить требования к техническим средствам, обосновать принципиальную возможность решения поставленной задачи. На третьем этапе этой стадии выполняют определение требований к программе, разработку технико-экономического обоснования разработки программы, определение стадий, этапов и сроков разработки программы и документации на нее, выбор языков программирования, определение необходимости проведения научно-исследовательских работ на последующих стадиях, согласование и утверждение технического задания.

2. *Эскизный проект*: разработка эскизного проекта, утверждение эскизного проекта.

На первом этапе идет предварительная разработка структуры входных и выходных данных, уточнение методов решения задач, разработка общего описания алгоритма решения задачи и разработка технико-экономического обоснования. На втором этапе — разработка пояснительной записки, согласование и утверждение эскизного проекта.

3. *Технический проект*: разработка технического проекта, утверждение технического проекта.

На первом этапе уточняют структуру входных и выходных данных, разрабатывают алгоритм решения задачи, определяют форму представления входных и выходных данных, определяют семантику и синтаксис языка, разрабатывают структуру программы, окончательно определяют конфигурацию технических средств, разрабатывают план мероприятий по разработке и внедрению программ. На втором этапе разрабатывают пояснительную записку, согласовывают и утверждают технический проект.

4. *Рабочий проект*: разработка программы, программной документации, испытания программы.

На первом этапе создают программу и выполняют ее отладку. На втором этапе разрабатывают программные документы в соответствии с требованиями действующих стандартов. На третьем этапе разрабатывают, согласовывают и утверждают программу и методику испытаний, проводят предварительные государственные, межведомственные, приемо-сдаточные и другие виды испытаний, корректируют программы и программную документацию по результатам испытаний.

5. *Внедрение*: подготовка и передача программы. Выполняют подготовку и передачу программы и программной документации для сопровождения и (или) изготовления, оформляют и утверждают акт о передаче программы на сопровождение и (или) изготовление, передают программы в фонд алгоритмов и программ.

Допускается объединять, исключать этапы работ и (или) их содержание.

ПО, как правило, разрабатывают в виде ПО для функционирования ЭВМ и ПО прикладного характера, включающее пакеты прикладных программ общего назначения (стандартные, типовые) и специального и (или) профессионального назначения. В 4-й главе они будут рассмотрены более подробно.

Для функционирования ЭВМ имеется пять классов *средств программирования*:

- редактирующие программы;
- транслирующие программы;

- программы-загрузчики;
- моделирующие программы;
- отладочные программы.

Редактирующие программы облегчают создание исходной программы и оперируют с ней, как с текстом, предоставляя различные возможности для изменений в тексте программы.

Транслирующие программы позволяют получить из исходной объектную программу, т. е. программу на машинном языке. Они реализуют программу, преобразуя запись на языке высокого уровня в последовательность машинных команд. Имеется два вида транслирующих программ: компиляторы и интерпретаторы, о которых будем говорить в теме 1.4.

Загрузчики переносят объектную программу из внешней памяти в оперативную память ЭВМ.

Моделирующие программы — межмашинные программы, позволяющие проверить (отладить) объектную программу без ЭВМ.

Отладочные программы облегчают отладку объектной программы на ЭВМ.

Контрольные вопросы

1. Что такое программы линейной структуры?
2. Что такое программы разветвляющейся структуры?
3. Что такое программы циклической структуры?
4. Каковы виды программных продуктов?
5. Каковы стадии и этапы разработки программных продуктов?
6. Какие программы являются средствами программирования?

1.4. Среда программирования. Редакторы.

Трансляторы. Отладка. Тестирование. Сопровождение

Программы создают (пишут) в определенной среде на ПЭВМ. *Среда* — это инструментальная программная оболочка, облегчающая составление и отладку программ. Чем совершеннее *среда программирования*, тем проще и быстрее создавать в ней новые программы.

Основные сведения о среде программирования рассмотрим на базе интегрированной среды Turbo Pascal. Система программирования (интегрированная среда) Turbo Pascal состоит из двух частей: компилятора с языка программирования Pascal и инструменталь-

ной программной оболочки, облегчающей составление и отладку программ.

Система Turbo Pascal — это совокупность программ и служебных файлов, записываемых на жесткий магнитный диск (ЖМД) в одном каталоге, обычно имеющем имя TP (или PAS, или TURBOPAS и т. п.). Для вызова следует отыскать в этом каталоге файл TURBO.EXE. Этот файл содержит готовую к работе диалоговую систему программирования. В него входят: текстовый редактор, компилятор, компоновщик и загрузчик. Для работы понадобится также основная библиотека, находящаяся в файле TURBO.TPL, и справочная служба — файл TURBO.HLP.

Например, для вызова системы по команде: D:\TP\TURBO в память персонального компьютера (ПК) загрузится программа TURBO.EXE и ей будет передано управление. На экране появится изображение основного окна программной оболочки (см. рис. 12). Для выхода из среды следует набрать команду ALT+X.

Интегрированная система программирования Turbo Pascal (IDE — Integrated Development Environment) во время работы содержит рабочее поле (desktop), на котором находятся окна с текстами программ на языке программирования, область меню, строка статуса и заголовок. В верхней строке экрана содержится меню возможных команд среды, в нижней — краткая справка о назначении основных функций клавиш. *Меню* — прямоугольный участок экрана, содержащий кодовые слова и предназначенный для диалога. Для выбора опции используется цвет и «ввод» или ALT+<буква>. Информацию о каждой опции можно получить, нажав клавишу F1. Каждое подменю содержит от 3 до 10 опций.

Окно — предназначено для обмена информацией между программистом и средой. Активное окно очерчивается двойной рамкой. *Диалоговое окно* — уточняет выбранные действия. В диалоговых окнах есть несколько полей. Для перехода от одного поля к другому используется клавиша TAB. Для перехода внутри поля используются клавиши перемещения курсора.

Кроме окна редактора, имеются окна отладочного режима, вывода редакторов работы программы, справок, стека и регистров. Их можно вызывать поочередно, или они могут присутствовать на экране одновременно.

Текстовый редактор — удобное встроенное средство создания и редактирования текстов программ. Окно редактора, очерченное двойной рамкой, предназначено для ввода и коррекции текста программ, в его верхней строке дается имя того дискового файла, откуда был прочитан текст программы (или имя NONAMEOO.PAS, если

текст не читается с диска). Имеется два специальных поля для работы с «мышью» (поля выделены квадратными скобками, а цифра в них — номер окна). Можно одновременно работать с несколькими программами (или частями большой программы) в разных окнах. Например, можно одновременно работать с 10 окнами редактора.

Выбор разделов в меню и пунктов подменю осуществляется указанием на них стрелкой и нажатием левой клавиши мыши. Для разработки программ самые нужные в подменю, выпадающим из раздела File, пункты New и Open.

New — создает незаполненное окно с пустой программой, которую можно после этого в нем составлять. *Open* — извлекает из памяти на жестком диске или дискеты уже имеющийся текст какой-либо программы. Появляется диалоговое окно, в котором нужно указать, какой именно объект (файл) следует открыть. Запоминание программы выполняется командой Save раздела File или клавишей F2. Для выполнения программы в среде IDE следует нажать клавиши Ctrl+F9 или команду RUN. Если компьютер не обнаружит ошибок, результаты появляются на пользовательском экране User screen, который находится за окном среды IDE. Потом снова появляются окна среды IDE. Результаты выполнения программы можно посмотреть с помощью клавиши Alt+F5. Нажав любую клавишу, можно возвратиться в окно Desktop, т. е. к окну программы.

Для перехода от состояния выбора режима из главного меню в состояние редактирования следует нажать Esc (ускользнуть, убежать), а для перехода к выбору из главного меню — F10. Редактор может раскрыть меню Edit. В этом меню содержится перечень команд для редактирования текста программ:

- Undo — отменить предыдущее действие;
- Redo — восстановить предыдущее действие;
- Cut — удалить в буфер;
- Copy — копировать в буфер;
- Paste — вставить из буфера;
- Clear — удалить.

Текст программы вводится с клавиатуры. Каждая строка завершается «вводом» (Enter), чтобы перевести курсор. Если курсор достиг нижнего края, то осуществляется «прокрутка» (смещение) окна редактора. Максимальная длина строки текста может быть 65 535 символов, как и вертикальная длина листа. Однако компилятор воспринимает строки программы не больше 126 символов. Окно можно смещать следующими клавишами:

- Pg Up — строка вверх;
- Pg Dn — строка вниз;

- Home — начало текстовой строки;
- End — конец текстовой строки;
- Ctrl+Pg Up — в начало текста;
- Ctrl+Pg Dn — в конец текста;
- Backspace — забой символа;
- Del — стереть символ;
- Ctrl+Y — стирает строку.

Тексты программ хранятся в *файлах* (см. тему 1.5). Если нет имени, то стандартное имя файла — NONAMEOO.PAS (см. рис. 12). Окно запроса выдает: Save file as (сохранить в файле с именем). В поле для ввода имени файла его нужно задать. Файл будет сохранен.

Трансляторы. Мы уже знаем, что для записи спецификации задачи в терминах некоторой модели процессора обработки данных на языке соответствующего процессора должны существовать программы, которые переводят команды языка программирования высокого уровня на машинный язык. Машина понимает только один язык: последовательности единиц и нулей. Программы, которые программисты пишут на различных языках, должны быть сведены к единому виду, т. е. машинному языку. Это достигается с помощью программ-трансляторов, написанных опытными профессионалами, как правило, системными программистами. Прежде чем выполнить программу, ее необходимо транслировать, т. е. обработать с помощью программы-транслятора. В результате трансляции создается исполняемый файл, он имеет расширение .exe. Трансляторы бывают двух видов: интерпретаторы и компиляторы.

Интерпретаторы — программы, анализирующие каждую инструкцию (каждый шаг) транслируемой программы отдельно, после чего преобразуют ее в машинный код и выполняют его. Если какой-либо фрагмент исходного кода повторяется, то интерпретатор будет снова анализировать и преобразовывать этот фрагмент перед его исполнением.

Компиляторы — программы, преобразующие в ходе непрерывного процесса весь файл исходного текста в машинный код, а затем осуществляют его выполнение. Если какой-либо фрагмент компилируемой программы повторяется, то компиляция повторно не производится. Компиляторы работают значительно быстрее, чем интерпретаторы, но для начинающих программистов легче обучаться на интерпретаторах.

Отладка программы. После подготовки текста программы ее можно попытаться выполнить, т. е. откомпилировать и связать (если необходимо) с библиотекой стандартных процедур и функций, загрузить в оперативную память и передать ей управление. Это

называется прогоном программы и реализуется командой RUN или клавишами Ctrl+F9. Если нет ошибок, то все действия выполняются последовательно. Компилятор может обнаружить ошибки в программе, которые следует исправить, но в этом случае действия прекращаются, восстанавливается окно программы, в верхней части которого красным цветом выводится сообщение об ошибке, а курсор устанавливается на строку программы, где она была обнаружена. Это позволяет ее исправить в тексте и отладить программу. Синтаксические ошибки можно быстро исправить с помощью команд меню Compile. Если ошибка математическая, то следует выполнить пошаговую отладку с помощью клавиш F4, F7, F8. Около переменной устанавливается курсор (Ctrl+F4), вводится имя переменной.

Тестирование (верификация) — проверка правильности работы программы. При тестировании программы используются специальные наборы параметров, для которых задача решается достаточно точно. Если полученный результат тестирования с этими параметрами удовлетворяет программиста и заказчика, то считается, что программа работает корректно.

Сопровождение. Анализ работы программы выполняется также и по результатам производственных испытаний, с наборами реальных данных. Поэтому после тестирования программы и передачи ее заказчику, следует обучить его работе с программой, дать консультации и устранить ошибки, выявленные в процессе производственной эксплуатации программы.

Контрольные вопросы

1. Что такое среда программирования?
2. Каковы функции редактора?
3. Каковы виды, назначение и функции трансляторов?
4. Как выполнить отладку программы?
5. Что такое тестирование и сопровождение программы?

1.5. Данные. Типы данных. Структуры. Хранение данных

Исходным объектом для программиста является **Задача**, сформулированная на содержательном уровне, т. е. описаны отношения между объектами реального мира. В любой момент времени объект находится в состоянии, которое характеризуется определенным набором свойств и их значений. Свойства объектов, сохраняющие

свое значение, являются постоянными, а изменяющие свое значение — переменными. Изменение свойств объекта происходит в результате выполнения заранее установленных действий. Изменению каждого свойства обычно соответствует определенная операция.

Программы манипулируют не с объектами, а с *данными*, т. е. информацией, представленной в форме, пригодной для ее передачи и обработки с помощью компьютера. Поэтому сначала надо определить данные, которые могут быть постоянными или переменными, их тип, и спроектировать структуры, описывающие реальные объекты и отношения.

Типы данных определяются множеством значений, которые могут принимать объекты данного типа (константы, переменные, выражения), и множеством допустимых операций над ними. Выделяют типы: *целый* (integer), *вещественный* (real), *логический* (logical) или *булевский* (boolean), *символьный* (char), *базовый* (byte). К структурным типам данных относятся: *массив* (array), *множество* (set) *строковый* (string), *запись* (record), *файловый* (file) и др.

Описание типов данных на языке Pascal приведено в теме 3.1. Целые (integer) могут принимать значения от -32 768 до +32 767. Поэтому для больших значений используют тип longint, в котором больше на 2 млрд значений. Типы числовых данных с дробной частью называются данными вещественного типа (real). Для записи чисел используется обычная форма. Но, если число слишком большое или маленькое (имеет большую разрядность), то используют экспоненциальную форму записи (с плавающей точкой).

Форма записи: $x = M \cdot E_{p_x}$, где M — мантисса, цифровая часть, E — основание системы счисления (для десятичных чисел $E=10$); p_x — порядок (степень), на который надо умножить M , чтобы получить число.

Примеры:

$1E-3 = 1 \cdot 10^{-3} = 0,001$; $0.257E+3 = 257$;
 $6.025 \cdot 10^{-23} = 6,025E-3 = 60,25e-24$.

```
Program Sarif_Sgeom;
var
  x,y :real;
  sa: real;
begin
  writeln ('Введите 2 числа');
  read (x, y); {Например, введем числа 6.9*E+2 и 2*E-2}
  sa: =(x+y)/2;
  writeln ('Среднее арифметическое= ', sa);
end.
```

Результат на экране: Среднее арифметическое=345,01

Перечисляемый тип данных.

Color = (white,blue,green,red) /color — цвет
 Toward=(links, rechts) / toward — обозначение направления
 Part = (west, nord, ost) /part — географическая часть света

Символы в качестве данных.

Пример.

```
Program prim Stroka;
var
  cm, ch: char;
begin
  writeln ('Введите произвольный символ');
  read (cm);
  writeln ('Вы ввели символ', cm);
  ch: ='Ж';
  writeln ('В символьную переменную занесен символ', Ж)
end.
```

Существует функция ord, позволяющая по значению символа определить соответствующий ему числовой код. Функция chr от аргумента целого типа дает значение символа, соответствующему числовому значению аргумента.

Логический тип данных. Во многих программах оказывается удобным предварительно вычислить, выполнено ли некоторое условие, далее пользоваться им в других частях программы, используя его сокращенное обозначение. Условия, вообще, являются логическими выражениями, которые могут принимать лишь значение ИСТИНА и ЛОЖЬ. Для абстрактного их представления в Pascal есть *логические переменные*, определяемые в разделе описаний служебным символом *boolean* (в честь математика Дж. Буля, который первым стал систематически рассматривать такие величины). Логические переменные можно задать константами *true* и *false*.

Простейшими вариантами логического выражения являются *логические константы*, *логические переменные* и *отношения* и определяются как запись:

Любое числовое выражение.

Арифметическое выражение _ Операция отношения _ Арифметическое выражение.

Операции отношения задаются как: =, <>, <, >, <=, >=.

Более сложные логические выражения получаются из простейших путем применения базовых логических элементов НЕ, И, ИЛИ (NOT, AND, OR) и круглых скобок.

Например: $(x < 4) \text{ and } (x < 7)$, $(x < 5) \text{ or } (x > 9)$,
 $(x > 0) \text{ and } (y > 0)$, $\text{not } (y \leq 8)$.

Задача. При указании возраста выводить утверждение: молодой (до 20 лет) и старый (после 60).

```
Program Vozrast;
var
  age: integer;
  young, old: boolean;
begin {использование логической переменной — X}
  write ('Введите свой возраст');
  read (age);
  young := age < 20;
  old := age > 60;
  if young then
    writeln ('Вы молоды');
  if old then
    writeln ('Вы старый');
end;
end.
```

Рассмотрим пример использования логических типов данных: True — «истина» и False — «ложь» при создании программ с подпрограммами.

Задача. Вводить и выводить цифры от 0 до 9 и буквы русского алфавита.

```
Program Alfavit;
Function IsDigit (ch: char): boolean;
begin
  if (ch >= "0") and (ch <= "9") then IsDigit := true
  else IsDigit := false;
end;
Function IsRBukwa(ch: char): boolean;
const
  bukwi: string = "А Б В Г... Я а б в г д е... я";
var
  k: integer
begin
  for k := 1 to length (bukwi) do
    if bukwi [k] = ch then IsRBukwa := true
  end;
var
  cc: char;
```

```
begin
  writeln ("Введите произвольный символ");
  read (cc);
  writeln;
  if IsDigit (cc) then writeln ("Введенный символ - цифра");
  if IsRBukwa (cc) then writeln ("Введите символ - русская буква")
end;
end.
```

Структуры данных. Процесс проектирования структур данных прост, если данные скалярного типа. В результате анализа задачи и рассмотрения возможных алгоритмов определяют структуру (сгруппированность и связь) используемых данных: скалярного типа, массивов (одномерные, многомерные) и т. д.

Массив — это набор однотипных данных, имеющий для всех элементов общее имя.

Пример. Карточка спортлото — объект реального мира, содержащий комбинацию из шести чисел:

```
12 18 23 26 41 42
 2  8 25 29 34 45
```

Здесь структурой данных будет *одномерный массив* из шести целочисленных компонент, содержащий определенную комбинацию чисел. Но может быть n комбинаций, а одномерный массив позволяет описать только одну. Если два одномерных массива, то целесообразно ввести двумерный массив, т. е. массив массивов: $\{a_{ij}, i = 1, n, j = 1, 6\}$. В карточке каждая строка массива содержит одну комбинацию из шести чисел, а всего в массиве n строк.

В математике числовые последовательности (матрицы, таблицы и др.) обозначаются как индексированные переменные.

Например:

a_n — член арифметической последовательности;

b_{ij} — элемент матрицы в i -й строке j -го столбца.

В языках программирования использование обозначений с индексами неудобно. Обобщением матричных понятий последовательности и матрицы для языков программирования служат *массивы*. Описание массива:

имя массива: *array* [диапазон 1], [диапазон 2] of_тип элементов.

При обращении к элементу массива его индекс указывают после имени массива в квадратных скобках. Индексом может быть не только число, но и выражение. Простейшие массивы — одномерные и соответствуют матричной последовательности (вектору).

Пример. Ввести последовательность из 10 целых чисел и вывести их в обратном порядке.

```
Program Massiv;
var
x: array[1..10] of integer; {массив из 10 целых чисел}
k: integer;
begin
  writeln ('введите 10 целых чисел последовательности');
  for k:=1 to 10 do {оператор цикла с параметром}
    read (x[k]);
  writeln; {переход на новую строку цикла}
  for k := down to 1 do
    write (x[k], ' ');
end.
```

В этой программе при определенном условии следует использовать break — оператор выхода из цикла. Например: `if x[k] = 99 then break.`

Многомерные массивы обобщают понятие матрицы.

Пример. Вычислить элементы матрицы $A_{ij} = i^2 + 2j$.

```
Program Mmassiv;
var
a: array [1..3, 1..4] of integer {двумерный массив из 3-х строк
                                и 4-х столбцов}
i: integer; {индекс строки}
j: integer; {индекс столбца}
begin
  for i:=1 to 3 do {внутренний цикл}
    for j:=1 to 4 do {внешний цикл}
      a[i,j] := i*i + 2*j; {формирование матрицы по известной
                           формуле};
    for i:=1 to 3 do
      begin
        for j:=1 to 4 do
          write (a[i,j]: 3, ' ');
        end;
      end;
end.
```

Тип множества. Этот тип задает интервал значений всех подмножеств, входящих в множество. Типом множества может быть любой скалярный тип. В Pascal его задают с помощью служебного слова *set*, например: `b: set of char`. Константы указывают в квадратных скобках, например: `digits=[0..9]`. Указание принадлежности элемента к множеству задают с помощью зарезервированного слова *in*, на-

пример: `if 2 in digits then...` Над множествами можно выполнять операции объединения, пересечения, разности и т. д., применять операции отношений (равенства `=`, неравенства `<>`, является подмножеством `<=`, является надмножеством `>=` и т. д.).

Строковый тип. Значения переменных строкового типа (*string*) — это строки символов. Например: `b: string [15]`. В квадратных скобках указывается длина строки. Допустимая длина — 255 символов. Строки можно объединять (операция конкатенации — с помощью знака `+` или функции `Concat(str1, str2)`). Можно преобразовывать числовые значения в строковые и наоборот с помощью процедур `Str(num, strnum)` и `Val(strnum, num, errcode)`, где `num` — значение числового типа, `strnum` — переменная строкового типа, `errcode` — сообщение об ошибке, если она есть. Второй переменной при переводе присваивается значение первой.

Тип записи. Тип «запись» — смесь разных типов. Этот тип данных имеет поля, которые могут содержать данные разных типов. Для описания переменных этого типа зарезервировано в Pascal слово *record*. Оно имеет следующий вид:

Имя переменной: `record` _ имена полей _ идентификаторы типов полей.

Например:

```
var
oblast: record
rang: integer;
les, lug, os: real;
```

Операции могут выполняться только над отдельными полями записи, но не над всей записью.

Файловый тип. Файлы могут быть текстовые, типизированные и нетипизированные. Текстовые файлы на языке Pascal имеют тип *text*. Параметром процедур работы с файлами является файловая переменная, которая устанавливает связь с файлом (см. далее «Хранение данных»). Имеется ряд стандартных процедур и функций для работы с файлами (например, `procedure Read (var fa: text; v1, ...)` считывает одно или несколько значений из файла, связанного с файловой переменной `fa`, в одну или несколько переменных `v1, ...`).

Типизированные файлы предназначены для хранения информации одного типа. Описание типизированного файла: `stabile : file of type_ID`, где `type_ID` — любой тип, кроме файлового.

Нетипизированные файлы используют для операций ввода-вывода, т. к. они дают прямой доступ к любому файлу на диске без

различия его типа и структуры. Описание нетипизированного файла: `untyped_file : file.`

Хранение данных. Данные, как правило, хранятся в файлах. *Файл* — это именованная совокупность данных, хранимая в памяти. Наиболее точное (уникальное) обозначение получается для файла, если в его наименование включать текст и информацию о месте хранения файла. Такое обозначение называют *полным именем файла*. Основная часть его имени в Паскале имеет не более 8 символов, а отделенное точкой расширение — не больше 3 символов. Для международного обмена через Internet рекомендуется включать в состав имен файлов только латинские буквы и цифры. Расширение имени файла обычно обозначает сокращение его назначения или содержания (.pas; myltext.txt и т. д.)

В любой файловой системе имеются определенные правила обозначения мест хранения файлов. В наиболее распространенных системах они имеют вид:

D:\цепочка_каталогов,

где D — обозначение логического диска; а далее — перечисление через специальный символ \ (слэш) названий каталогов, через которые нужно пройти, чтобы от корневого каталога логического диска добраться до места хранения файла. Например: `c:\bp\work\myltext.txt`. Один из каталогов является текущим в каждый момент времени.

Каталог при создании называется именем и предназначен для хранения информации о файлах, а не самих данных. Передвигаться по каталогам можно с помощью простейших команд ОС.

Файл внутри программы принято называть специально вводимой *файловой переменной* или *указателем*. Файлы подразделяют на бесструктурные, текстовые и типизированные (на языках программирования).

Базовые операции с файлом. Помещение данных в файл называется *записью* в файл, а извлечение хранимых в нем данных — *чтением* из файла. Подготовительные операции: открытие, закрытие файла.

В языке Паскаль открытию предшествует подготовительная операция связи (оператор *assign*) между именем файла и его условным обозначением в программе — файловой переменной (ФП):

```
assign (имя_ФП, имя_файла)
Имя_ФП: file of _ тип элементов файла
```

Открытие файла выполняется оператором *rewrite* или *reset*.

Закрытие файла выполняется оператором *close*.

Пример. Создать файл с именем `primera.num`, в котором запоминается последовательность из 12 целых чисел, заданных простой расчетной формулой.

```
Program Posledov;
var
fa: file of integer;   {fa — файловая переменная}
k, nn: integer;
begin
  assign (fa,'primera.num'); {файл типизированный}
  rewrite (fa);             {для записи в файл}
  for k:=1 to 12 do
  begin
    nn=k*k+1;
    write (fa, nn);
  end;
  close (fa);
end.
```

Произвольный *доступ* к данным в файле может быть *последовательным* и *прямым*. При первом для прочтения какого-либо его элемента следует последовательно прочитать все предшествующие ему элементы с начала файла. При прямом доступе можно явно задать место в файле, где находится требуемый элемент, и прочитать его. В массивах элементы нумеруются явно (индексы указывают элемент массива). В файлах нумерация элементов неявная, поэтому следует использовать *указатель* текущей записи в файле. Этот указатель поддерживается файловой системой и прикладному программисту недоступен. В языках программирования есть специальные процедуры изменения указателя текущей записи в файле. Значение указателя текущей записи задается величиной длинного целого типа (`longint`). Процедура установки текущего указателя в Паскале обозначается *seek* (например, `seek (fb, pos-1)`); {переход от `pos` к `pos-1`, чтобы отсчет шел от 0}}.

Текстовые файлы возникли, чтобы сохранять для дальнейшего использования изображения текстов на экране, и являются аналогом текстового издания на экране, создаваемого операторами вывода. Текстовые строки, завершаемые невидимыми служебными символами: (13) — стандартный символ CR — carriage return (возврат каретки) и (10) — стандартный символ LF — line feed (перевод строки). Для операционных систем в IBM PC это осуществляет клавиша Enter.

Текстовый характер ввода или вывода должен быть указан при открытии файла. В Паскале это может быть неявно.

Пример. Вывод в текстовый файл на языке Паскаль

```
program Vtext;
var
fa: text; {имя_ФП: text};
k, na: integer;
begin
  assign (fa,'primerc.txt');
  rewrite (fa);
  for k:=1 to 1
  writeln (fa, 'квадрат числа', k, 'равен', k*k, 'а куб-', k*k*k);
  close (fa);
end.
```

Контрольные вопросы

1. Что такое данные и каковы их виды?
2. Что такое тип данных?
3. Какие типы данных используют при программировании на языке Паскаль?
4. Что такое структуры данных?
5. Как осуществляется хранение данных?

Глава 2

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

2.1. Классификация языков программирования и этапы их развития

Язык программирования — совокупность основных символов (алфавит) и правил составления из них смысловых конструкций. Он должен обеспечивать взаимодействие человека с машиной, быть удобен и понятен им обоим.

В соответствии с представлениями об уровне абстракции, на котором формируется задача и процессы обработки данных в реальном устройстве (процессоре), говорят об *уровне* языков программирования. Можно выделить языки низкого и высокого уровня.

К языкам низкого уровня относят:

- машинные языки — языки кодов ЭВМ;
- машино-ориентированные языки — ассемблеры, мнемокоды.

К языкам высокого уровня относят:

- проблемно-ориентированные (имеют средства для организации структур данных, описания алгоритмов и ориентированы на решение задач определенного класса): Фортран, Алгол, Кобол, Ада и др.;
- универсальные: Алгол 68, PL/I, Паскаль, QBasic, C++, C# и др.;
- языки проектирования программ (системы программирования) — в настоящее время имеют самый высокий уровень абстракции, они расширяются не как языки описания процесса обработки данных, а как средства описания задач: *Visual Basic, Delphi, MS Visual C++, Borland C++ Builder* и др.;
- языки гипертекстовой разметки, такие, как HTML — набор кодов, который вводится в документ для обозначения, например, связей между его частями. Команды HTML обеспечивают соединение сайтов и главных страниц WWW (Всемирной паутины сети Интернет) при помощи гиперссылок и указыва-

ют Web-браузеру (программе навигации) способ расположения массивов данных;

- языки описания сценариев — макросы, в которых объединены отдельные команды, управляющие средой в соответствии с их списком — программой: (например состоящие из именованных последовательностей совокупности указанных нажатий клавиш при работе с пакетом Microsoft Office);
- языки моделирования систем: например, GPSS (General Purpose Simulating System) позволяет автоматизировать при моделировании процесс программирования моделей. Язык построен в предположении, что моделью сложной дискретной системы является описание ее элементов и логических правил их взаимодействия. Для определенного класса моделируемых систем выделяют небольшой набор абстрактных элементов-объектов. Набор логических правил ограничен и может быть описан небольшим числом стандартных операций. Комплекс программ, описывающих функционирование объектов и выполняющих логические операции, является основой для создания программной модели систем данного класса.

Языки реально функционирующих процессов называются *языками низкого уровня* (например язык для функционирования процессора). Написание программ на языках низкого уровня требует более высокой квалификации программиста. Первые языки программирования представляли собой машинные коды, они были эффективны и удобны для работы ЭВМ, но весьма трудны для восприятия человеком и требовали от него длительного изучения и специальных навыков использования. Программист должен был писать действительные адреса ячеек памяти, в которых размещены данные, участвующие в операциях, результаты, а также адреса команд.

Настоящим языком программирования считают созданный для первых ЭВМ в 1949 г. под руководством Мочли язык *Short Code* (*Короткий код*), с помощью которого можно было записывать уравнения не в двоичном коде, а с помощью двухсимвольных комбинаций.

Язык *Assembler* (*Ассемблер*) — машинно-ориентированный, использующий символическое представление машинного языка, операции которого записываются в форме, более понятной программисту, применяются мнемоники. Имеется возможность объединения нескольких машинных команд в макрокоманды. Например, вместо перечисления всех команд в программе можно записать имя макрокоманды. Но программы на этом языке громоздки, их написание сложно, но для задач, связанных с аппаратной частью ЭВМ, они эффективны. Поэтому для решения указанных задач вставки на Ас-

семблере до сих пор используют в программах на языках более высокого уровня.

Потребность в более понятных и удобных языках для написания программ различного назначения привела к созданию языков высокого уровня.

Языки высокого уровня максимально приближены к естественному человеческому языку, что значительно упрощает их изучение и использование. В языках высокого уровня инструкции, написанные программистами, зачастую выглядят как обычный текст на английском языке с применением общепринятых математических знаков. Для реализации программ, написанных на языках высокого уровня, как правило, создают *программы-компиляторы* (разновидность трансляторов), которые вводят «исходный код» (специальные символы, слова и математические выражения), преобразуют его в «объектный код», подключают с помощью редактора связей необходимые стандартные программные модули из библиотек и заставляют компьютер выполнять соответствующие команды на машинном языке («исполняемый код» — рабочая программа).

В конце 1950-х гг. появились три языка: Фортран, Алгол и Кобол, положившие начало эпохе развития языков программирования высокого уровня.

В 1954—1957 гг. на фирме IBM (США) группой инженеров под руководством Джона Бэкуса был разработан язык программирования *Fortran* (Formula Translation), предназначенный для решения математических и инженерных задач. Язык прост в освоении, а потери в эффективности, по сравнению с Ассемблерами, были минимальны. В языке отсутствовало строгое форматирование текста по столбцам, допускались пробелы в тексте, операторы в строке могли располагаться произвольно, математические формулы записывались почти в их обычном виде. Программирование на языке стало доступно многим.

В 1958 г. появляется первая версия языка *ALGOL* (ALGOrithmic Language), принятая на совещании в Цюрихе (Швейцария), в котором принимали участие ведущие специалисты из США, в их числе Джон Бэкус. Язык предназначался для записи алгоритмов вычислительных задач. Он стал первым языком с блочной структурой.

В 1959 г. для решения коммерческих задач группой производителей и пользователей компьютеров был создан язык *COBOL* (Common Business Oriented Language) — универсальный язык, предназначенный для бизнеса. Язык хорошо структурирован. Cobol-программа состоит из четырех самостоятельных, следующих в строго определенном порядке разделов: идентификация (задает название

программы и содержит справочную информацию); оборудование (приводятся параметры ЭВМ); данные (описывает обрабатываемые данные); процедуры (операторы).

В 1964 г. объединенной комиссией фирмы IBM и разработчиками проекта Share Fortran был предложен язык *PL/I* (*Programming Language One*), вобравший в себя черты Фортрана, Алгола-60 и Коболы. Язык совершенствовался до 1966 г., был пригоден для описания алгоритмов научно-технических, экономических, информационно-логических задач и стал универсальным языком широкого применения. Программа на *PL/I* состоит из последовательности операторов двух групп: исполняемые (ввод-вывод данных, логические и расчетные операции, присваивание вычисленных значений некоторым переменным и др.) и неисполняемые (для описания и передачи транслятору информации о свойствах определенных объектов программы). Но написание текста программы на этом языке было достаточно неудобно, т. к. структура текста осталась еще с тех времен, когда программы вводились в ЭВМ с помощью перфокарт (на специальных бланках, имеющих 80 позиций в строке). Кроме того, большое количество средств и разнообразие операторов привело к тому, что *PL/I* оказался довольно сложным в изучении.

В 1961—1965 гг. Джоном Маккарти в Массачусетском технологическом институте был создан язык функционального программирования (программа описывает вычисление некоторой функции), который ориентирован на решение задач нечисленного характера. *ЛИСП* (*LISP — List Processing Language*) — символьный язык для обработки списков. Понятие «список» понимается широко и включает представление алгебраических выражений, графов, элементов конечных групп, множеств, правил вывода и многое другое. Основные типы данных в языке называются «атом» и «точечная пара». *ЛИСП* создавался как язык для исследований по проблеме искусственного интеллекта. Основой этого языка служит математическая теория алгоритмов и *рекурсивных функций*, т. е. вычисление значения функции через значения этой же функции от других элементов. Присваивания и циклов в функциональном языке нет. *ЛИСП* — универсальный язык, т. к. любой алгоритм может быть описан с помощью некоторого набора рекурсивных функций, что позволяет моделировать на ЭВМ сложные алгоритмы обработки данных, в том числе алгоритмы моделирования интеллектуальной деятельности людей. В 1984 г. вышел *Common Lisp*, а затем система *Common Lisp Object System (CLOS)*. В дальнейшем были разработаны программные продукты, основанные на *ЛИСП*: *Common Lisp*, *Mac Lisp*, *Inter Lisp*, *Standard Lisp*, *Common Loops* — система для обработки знаний

и программирования, *New Flavors* — система посылки сообщений, используемая в коммерческих целях, и др.

В 1965 г. в Институте кибернетики АН Украинской ССР под руководством академика В. М. Глушкова для ЭЦВМ «Мир» был разработан язык высокого уровня *Алмир* (Алгоритмический язык для машины «Мир»), транслятор с которого был постоянно встроен в машину. Язык был разработан на базе Алгола, но использовал символику русского языка и стал одним из первых (возможно, первым) национальным языком программирования.

В 1966 г. в Институте прикладной математики АН СССР В. Турчиным был создан функциональный язык *РЕФАЛ* (*алгоритмический язык рекурсивных функций*). Построенный на алгоритмах М. Маркова, он удобен для обработки текстов. Язык активно использует мощные средства преобразования списков на основе концепции распознавания по образцу.

В 1970-е гг. появилось непроцедурное (декларативное) программирование — программирование на функциональных и логических языках, начавшее быстро развиваться в 1980—1990-е гг. в связи с разработкой в Японии проекта создания ЭВМ V-го поколения — поколения интеллектуальных машин.

Программа на логическом языке вообще не описывает действия. Она задает данные и соотношения между ними. После этого можно задавать вопросы. ЭВМ перебирает известные (заданные в программе) данные и находит ответ на вопрос. Порядок перебора не описывается в программе, а неявно задается самим языком.

Классическим языком *логического программирования* считается *ПРОЛОГ* (*Prolog — Programming in Logic*) — язык для создания систем искусственного интеллекта, разработанный в 1971—1972 гг., сотрудником университета в Лумини (Франция) Аланом Колмари. Пролог (*Пролог++*) — символьно-логическая система программирования, вначале предназначенная для решения теорем, сейчас используемая для поиска решений, связанных с искусственным интеллектом. Язык позволяет в формальном виде описывать различные утверждения, правила рассуждений, заставляет ЭВМ рассуждать и давать ответы на заданные вопросы. Программа состоит из некоторого множества отношений, а ее выполнение сводится к выводу нового отношения на основе заданных. Понятия и принципы языка основаны на понятиях математической логики и аппарата автоматического доказательства теорем, созданного в ходе исследований по «искусственному интеллекту». У языка есть потомки: *Parlog* (ориентирован на параллельные вычисления, выпущен в 1983 г.), *Delta* и др.

С течением времени модифицировался Фортран, появлялись его новые версии (1958 г. — Фортран-II, 1961 г. — Фортран-III, 1962 г. — Фортран-IV, 1966 г. — Фортран-66, 1977 г. — Фортран-77, 1984—1988 гг. — Фортран-88). Особенно широко использовалась версия Фортран-IV. В 1966 г. Американская организация стандартов (впоследствии преобразованная в Американский национальный институт стандартов — ANSI) издала первый стандарт программирования, известный как Fortran-66. В 1977 г. вышел новый стандарт — Fortran-77, расширивший возможности языка, особенно для обработки текстов и работы с файлами. В 1991 г. появился стандарт — Fortran-90, дополненный некоторыми расширениями. Последним стандартом языка был Fortran-95. Он и в настоящее время применяется для решения инженерных и научно-технических задач. Появился и Visual Fortran.

В 1965 г. на основе Фортрана профессора Дартмутского колледжа (США) Джон Кемени и Томас Куртц (Kurtz) для обучения студентов разработали язык *BASIC* (Beginner's All-Purpose Symbolic Instruction Code) — универсальный символьный программный код для начинающих. Язык прост и в то же время имеет широкие возможности для обработки символьной и графической информации. Его характерные черты: диалоговый режим работы, нумерация строк, вещественный и символьный типы данных, управляющие конструкции, все переменные являются глобальными, наличие массивов. Язык нашел широкое применение и имеет много версий. Одну из них написал Билл Гейтс — основатель корпорации «Microsoft». В 1980-х гг. эта корпорация выпустила *QuickBasic*, обладающий большим быстродействием, простотой, компактностью. Но основная часть языка во всех версиях одна. Вот лишь несколько версий языка: *Basic*, *GWbasic*, *Power Basic*, *Turbo Basic*, *QuickBasic* (компилирующая система языка входит в состав MS-DOS, начиная с версии DOS 5.0), *Visual BASIC* (для программирования в среде Windows).

В 1960 г. на конференции в Париже представители США, ФРГ, Великобритании, Франции, Дании, Голландии, Швейцарии утвердили улучшенную версию языка АЛГОЛ — АЛГОЛ-60, получившую широкое распространение. На ее основе в 1960-е гг. были разработаны такие языки, как *Simula-67* — первый объектно-ориентированный язык, в котором впервые были определены понятия объекта, класса и наследования; АЛГОЛ-68 — с усиленными средствами ввода-вывода, но оказавшийся слишком громоздким и не получивший широкого признания; в 1970-е гг. — алголоподобные языки Паскаль, Ада, Си; в 1980-е — C++; к концу 1990-х — C# (Си шарп).

Среди разработчиков АЛГОЛ-68 был швейцарский ученый Николаус Вирт, разработавший в 1970 г. язык *Pascal* для новой операционной системы UNIX, установленной на компьютере CDC 6000. Язык нашел широкое применение и явился базой для создания Pascal-подобных языков: *Ada*, *Modula-2*, *Oberon* — два последних были разработаны также Н. Виртом.

Язык *Ада*, названный в честь Августы Ады Лавлейс, разработан группой французских ученых под руководством Ж. Ишбиа в период 1975—1980 гг. по заказу министерства обороны США для создания бортовых систем управления военными объектами. Конечная версия международного стандарта языка была опубликована в 1987 г. Последний стандарт описывает версию языка Ада-95. Она является первой в мире объектно-ориентированной системой программирования, на которую был введен международный стандарт — ISO/IEC 8652:1985 (E). Структура Ады очень похожа на Паскаль и имеет возможность использовать неоднородные структуры, разделение памяти, реализовать неядные функции преобразования типов. Язык ориентирован на тщательный контроль программ, чтобы надежность военных систем была максимальной. Ада располагает большим набором данных (например, *ch* — символьный, такого типа нет ни в Фортране, ни в Алголе-60), возможностью строить новые типы данных, не предусмотренные ранее. Язык оказался достаточно универсальным и применяется не только для решения военных задач.

Язык *Модула-2* создан в 1979 г. Н. Виртом. В языке сочетается простота Паскаля и средства модулизации. Главное свойство языка — хорошо продуманная структура программ. Характерная черта — раздельная компиляция, позволяющая разрабатывать и хранить в библиотеках программы, которые можно использовать повторно. Язык прост для изучения и эксплуатации.

Язык *Оберон* создан Н. Виртом в 1987 г. как попытка достичь идеала универсального языка программирования. В нем имеются средства обогащения комбинированных типов данных, средства объектно-ориентированного программирования — расширяемые записи. Язык обеспечивает строгий контроль на этапе трансляции. В 1992 г. Х. Мессенбек предложил расширенную версию языка — Оберон-2. В настоящее время семейство оригинальных Оберон-систем известно под названием *Oberon V4*.

Язык *C* (*Ci*) создан Д. Ритчи и Б. Керниганом в 1972 г. В 1979 г. Бьерном Струострупом разработан «Си с классами», а в 1983 г. — C++. Этот язык в основном является языком Си со специальными синтаксическими расширениями для определения и управления объектами, кроме того введена поддержка абстракции данных, про-

верка типов аргументов функций и макроподстановка функций. Язык, повсеместно эффективно использующийся для объектно-ориентированного программирования, постоянно совершенствуется, а его автор, возглавляющий до настоящего времени отдел программных исследований в лаборатории AT&T Labs, работает в области технологии распределенных и операционных систем, имитационного моделирования, проектирования и программирования.

Были разработаны такие языки, основанные на C, как Objective-C, C-talk, Complot C.

В настоящее время существует несколько тысяч языков и систем программирования. Однако широкое распространение получили лишь несколько десятков из них.

Наиболее популярными в настоящее время являются языки проектирования программ — *системы визуального программирования*, имеющие самый высокий уровень абстракции. Они являются самым современным и эффективным средством создания программных продуктов и рассматриваются не как языки описания процесса обработки данных, а как средства описания задач. Среди самых популярных можно назвать системы программирования: *Visual BASIC, Delphi, C++Builder, Visual C++*, базирующиеся соответственно на языках *BASIC, Pascal* и *C++*. Как правило, они имеют интегрированную среду разработки IDE, включающую текстовый редактор, конструктор форм и набор других инструментов, позволяющих значительно сократить объем работ по созданию программных продуктов.

В 1991—1993 гг. «Microsoft» создает первую визуальную среду программирования (систему программирования) *Visual BASIC*, позволяющую достаточно просто создавать MS Windows-программы без особого внимания к сложной структуре этой операционной системы. *Visual BASIC* обладает средствами визуального проектирования, упрощает программирование на языке *BASIC* и использует все графические функции Windows. Он стал одним из первых, поддерживающих событийно-управляемое программирование (event-driven programming), позволяющее программисту не описывать каждый шаг, а только указать, как реагировать на различные события (выбор команды, движение или щелчок мыши и т. д.), создавая таким образом набор управляемых взаимодействующих процедур. *Visual BASIC* связан с приложением Windows — Microsoft Office, он снабжен встроенными средствами поддержки баз данных (БД) Access.

Отдельно следует выделить язык *dBase* фирмы «Borland», который представляет собой гибрид объектно-ориентированного и традиционных языков, применяющийся для реализации БД. При их

создании реализуют объектный дизайн и обычные приемы обработки записей. Был создан целый ряд эффективных *dBase*-подобных систем управления БД (СУБД) FoxBase, Paradox и др.

В 1995 г. фирма «Borland» создает систему программирования Delphi, а фирма «Microsoft» — Visual C++.

Языки описания сценариев — подвид пакетных файлов — подобны макросам, в которых объединены отдельные команды, управляющие операционной средой в соответствии со списком, являющимся программой. Основное назначение этих языков — связывать различные компоненты и приложения друг с другом. Здесь нашел применение безтиповой подход к описанию данных, что значительно упрощает установление связей между компонентами и ускоряет разработку прикладных программ. Эти языки предназначены для комбинирования компонентов, набор которых создается заранее при помощи графического интерфейса пользователя, Internet или инфраструктуры разработки и технологии сценариев. Например, языки Perl, Tcl, Active X, Java и др.

PERL (Practical Extraction and Report Language) — язык для практического извлечения данных и составления отчетов, а также для обработки произвольных больших текстов и файлов. Его часто используют при написании системных программ. С помощью языка можно не только сканировать текстовые файлы, но и обрабатывать двоичные данные. Синтаксис языка близок к синтаксису Си. Язык позволяет использовать регулярные выражения, создавать объекты, вставлять фрагменты программ в программы на C и C++, осуществлять доступ к базам данных. С помощью языка можно взаимодействовать с Web-серверами, получать данные из формы HTML.

Популярность объектно-ориентированного языка C++ дала базу для создания множества новых языков для современного Internet. С развитием глобальной сети Internet появилась проблема переноса программ с одной платформы на другую. Для ее решения и создания мобильных сетевых программных продуктов фирмой «Sun Microsystems» в 1991 г. был создан новый язык *Java*. Язык разработали Д. Гослинг, П. Нотон, К. Ворт, Э. Фрэнк, М. Шеридан и первоначально называли его ОАК. *Java* позволяет описывать любые задачи, включая обслуживание Web-страниц, по синтаксису и технологии программирования очень похож на C++. Программы, написанные на языке *Java*, независимы от архитектуры ЭВМ. Это достигается путем транслирования программ в промежуточный язык, называемый байт-кодом. *Java* имеет защиту от компьютерных вирусов и несанкционированного доступа. Главным отличием *Java*-программ, которые называются *Java-applications* (апплеты), является использо-

вание библиотеки Java-классов, обеспечивающих разработку безопасных распределенных систем. В 2003 г. компания «Sun» объявила о появлении нового продукта Joe, предназначенного для существенного облегчения встраивания Java-клиентов в информационные системы Internet.

Для целей мобильного программирования фирма «Netscape» создала язык LiveScript, позволяющий простые программы включать в текст документов, написанных на HTML. Затем эта фирма с разрешения фирмы «Sun» переименовала LiveScript в *JavaScript*, с помощью которого удобно создавать сценарии, связанные с определенными событиями.

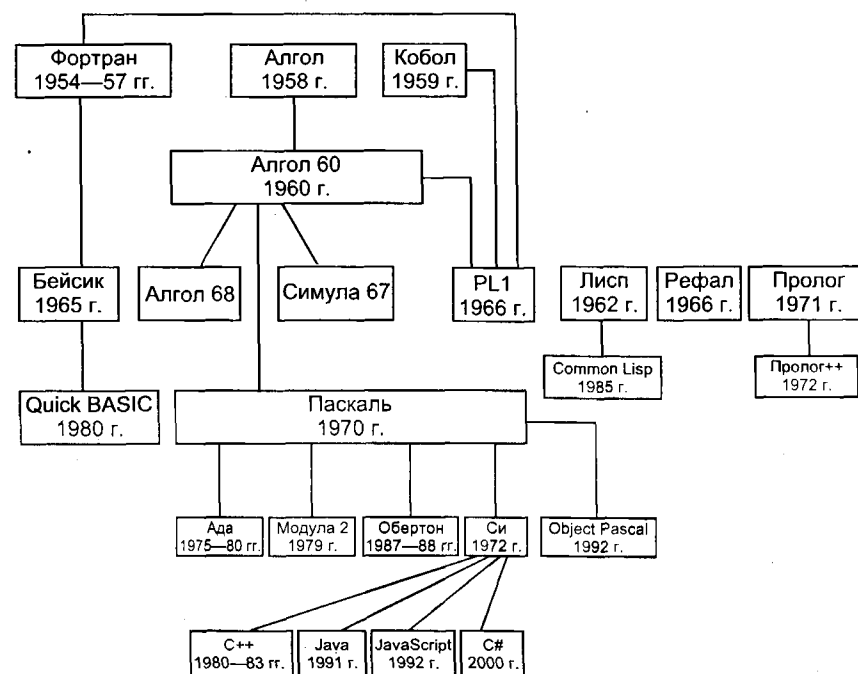
HTML (Hiper-Text Markup Language) — язык гипертекстовой разметки, необходимый для программирования во Всемирной паутине — WWW (Word Wide Web), а также для написания программного кода для Web-страниц. Первую версию HTML разработал Тим Бернерс-Ли — сотрудник Европейской лаборатории физики элементарных частиц. По существу, это язык форматирования, определяющий, как на Web-странице будут размещены текст, графика, таблицы. Можно также создавать ссылки между различными частями информации. Основу HTML представляет обычный текст, в который вставляются управляющие символы (тэги): получаем Web-страницу. Этот объект может содержать информацию различного вида, просмотреть которую можно с помощью специальных программ-браузеров (Internet Explorer, Netscape Navigator и др.) Найти Web-страницу можно с помощью универсального указателя ресурсов (URL — Universal Resource Locator). Для доступа к Web-странице используется протокол передачи гипертекста HTTP (Hiper Text Transfer Protocol), т. е. сначала указывается протокол, затем имя сервера и путь к файлу Web-страницы.

SQL — непроецедурный язык высокого уровня, чем принципиально отличается от традиционных алгоритмических языков. Он представляет собой субязык данных, используемый для создания интерфейса БД и реализован в популярных СУБД, а также стал стандартным языком запросов. В 1992 г. он объявлен международным стандартом. SQL включает язык определения схемы (SQL-DDL) и язык манипулирования данными (SQL-DML). Язык позволяет задать только то, что нужно делать, а само исполнение отдельных операций возложено на СУБД. Особенностью этого языка является так называемая трехзначная логика. Кроме булевых значений: истина и ложь, выражение может принимать значение NULL (пустое значение) — специальный код, который используется при отсутствии данных. В операциях сравнения NULL используется, если результат не-

известен, и может применяться в одной из трех форм: интерактивный, или автономный, SQL (для непосредственного извлечения информации из базы или записи в нее); статический SQL (для записи фиксированного исполняемого кода SQL); динамический SQL (для генерирования кода SQL во время исполнения приложения, построения запросов в диалоговых средах, в графических средствах разработки приложений БД).

В конце 1990 г. на фирме «Microsoft» под руководством Андерса Хейльсберга был разработан язык C# (Си-шарп). Альфа версия выпущена в 2000 г., а в 2001 г. — окончательная версия вошла в пакет Visual Studio.Net 7. Язык унаследовал лучшие свойства языков C и C++. C# работает в среде исполнения .NET Framework, что обеспечивает переносимость программ в многоязыковое программирование. Программы на C# могут быть скомпилированы в среде Visual C++.

Таким образом, хронологию развития наиболее распространенных языков программирования можно представить в следующей последовательности.



В настоящее время программирование на персональных ЭВМ ведется в основном на языках BASIC, Pascal и C (как правило, их последних версиях), а на больших и мини-ЭВМ все еще применяются языки Фортран и ПЛ/1. Что касается языков программирования для перспективных ЭВМ нового поколения, то прототипом будущего языка считается Пролог.

Для того чтобы иметь представление о записи программ, написанных на разных языках, посмотрим их тексты на трех языках: BASIC, Pascal и C++ для решения одной и той же задачи — вычисление корней квадратного уравнения. Программа решения этой задачи на языке Паскаль приведена также в теме 1.2.

Строки в программе Basic нумеруются по возрастанию (как правило, числами кратными 5 или 10), что удобно при использовании оператора безусловного перехода (goto). Язык имеет небольшой набор типов данных: целые (от -32 768 до +32 767), строковые (до 256 символов), с плавающей точкой обычной точности (7 цифр), с плавающей точкой двойной точности (16 цифр). В Basic есть средства обработки многомерных массивов, логические операции, управление форматом выводимых данных, обработка графики и др.

BASIC

```

10 PRINT "Вычисление корней квадратного уравнения"
20 PRINT "Введите коэффициенты уравнения"
30 INPUT a
40 INPUT b
50 INPUT c
60 d = b*b - 4*a*c;
70 IF d<0 GOTO 120
80 PRINT "Корни уравнения"
90 x1 = (-b + sqrt(d))/(2*a) : x2 = (-b -sqrt(d))/(2*a)
100 PRINT x1, x2
110 GOTO 130
120 PRINT "Уравнение не имеет корней"
130 END

```

PASCAL

```

Program KUrPas;           {Заголовок программы}
Var                       {Описание переменных}
a,b,c: real;              {Коэффициенты уравнения}
d,x1,x2: real;            {Вспомогательные переменные}

```

```

begin                     {Начало программы}
  write('Вычисление корней квадратного уравнения');
  writeln('Введите коэффициенты уравнения');
  read (a);
  read (b);
  read (c);
  d = b*b - 4*a*c; {Дискриминант}
  if d<0 then write('Корней нет')      {если d<0, то вывод на экран:
                                         {Корней нет}
  else                                {иначе}
begin                               {Начало серии команд}
  x1 = (-b + sqrt(d))/(2*a);          {Вычисление корней}
  x2 = (-b -sqrt(d))/(2*a);
  write ('Корни квадратного уравнения x1 =', x1; 'x2 =', x2)
end                                  {Конец серии}
end                                  {Конец программы}

```

C++

```

# include<iostream.h>      /* сообщение компилятору, что он должен
                             /* включить стандартные программы
# include<math.h>           /* из библиотек
# include<conio.h>
main ( )                    /* функция без аргументов,
                             /* ее инструкции заключаются далее в { }
{
  int a, b, c, d, x1, x2;
  cout <<"Вычисление корней квадратного уравнения";
  cout <<"Введите коэффициенты уравнения";
  cin >>a;
  cin >>b;
  cin >>c;
  d = b*b - 4*a*c;
  if (d<0) goto MET1;
  cout <<"Корни уравнения";
  x1 = (-b + sqrt(d))/(2*a);
  x2 = (-b -sqrt(d))/(2*a);
  cout <<"x1<<x2;
  getch ( );
  goto MET2;
MET1:
  cout <<"Уравнение не имеет корней";
  getch ( );
MET2:
}

```

Контрольные вопросы

1. Что такое язык программирования?
2. Какие языки относят к языкам низкого уровня? Приведите примеры.
3. Какие языки относят к языкам высокого уровня? Приведите примеры.
4. Каковы основные характеристики языка гипертекстовой разметки HTML?
5. Что такое система визуального программирования? Приведите примеры.

2.2. История развития языков программирования Паскаль и Си

Язык программирования Pascal (*Паскаль*) создал в 1970—1971 гг. Никлаус Вирт — профессор Института информатики Высшей политехнической школы Швейцарии. Он разработал этот язык для обучения студентов навыкам программирования. Язык назван в честь французского философа и математика XVII в. Блеза Паскаля, который изобрел вычислительное устройство. Язык быстро получил широкое распространение, т. к. был легок для изучения, четок и логичен, а программы на нем — наглядны и хорошо читаемы.

Язык вобрал в себя все лучшие решения языка Алгол 60, но он стал новым языком, в котором впервые были введены абстрактные типы данных; т. е. введены средства конструирования новых типов данных, интервальные типы данных, средства работы со множествами и др. По сравнению с Алголом он имеет небольшое число операторов и других конструкций, однако позволяет писать достаточно сложные программы, включает средства обработки трехмерной графики. Хотя в языке есть и недостатки — например невозможность использования динамических массивов переменной длины, — Паскаль стал первым языком, в котором четко реализованы принципы структурного программирования. Многие языки, появившиеся позднее, содержат аналогичные Паскалю структуры. На его базе созданы Ада и Модуль-2. С этим языком, как правило, знакомятся большинство будущих программистов в мире, т. к. он стал интересным и мощным средством программирования прикладных задач и средством обучения программированию как дисциплине, основанной на понятиях, четко отраженных в Паскале. Язык является универсальным, ориентированным на решение задач любого класса.

К 1979 г. в Паскаль группой авторов были внесены изменения, и он был опубликован в окончательном варианте. На язык программирования в 1982 г. появился стандарт ИСО7185 (ISO) — Международная организация по стандартизации.

Трансляторы для программ, написанных на Паскале, разработаны для различных компьютеров и в настоящее время имеют много разновидностей. Первый транслятор с языка был создан в 1973 г. С этого времени язык стал популярен.

Существует много версий языка Паскаль. Базовая версия Н. Вирта имеет гораздо меньше возможностей, чем мощная профессиональная версия системы программирования Турбо Паскаль (Филипп Кан, Андерс Хейлсберг, фирма «Borland»), состоящая из компилятора с языка Паскаль и инструментальной оболочки, способствующей повышению эффективности создания программ. Но версии совместимы, т. е. любая программа «младшей» версии соответствует «старшей».

В 1982, 1985 и 1988 гг. последовательно появляются версии Турбо Паскаль 2.0, 3.0, 4.0, в которых постепенно наращиваются возможности быстрой компиляции, удобства и другие возможности системы. В 1988 г. выходит версия 5.0 с хорошо развитым программным окружением, в том числе встроенным отладчиком программ. К настоящему времени уже имеется версия Турбо Паскаль 7.0. Турбо Паскаль позволяет применять приемы объектно-ориентированного программирования и может использоваться в большинстве существующих для ПК ОС.

В 1991 г., сразу после появления Windows 3.1 фирма «Borland» для разработки программ в среде Windows выпускает Turbo Pascal for Windows, а в 1992 г. его усовершенствованную версию Borland Pascal with Objects. В 1995 г. фирма выпускает первую версию системы программирования Delphi, затем с интервалом в год — еще четыре версии, а в 2001 г. выходит шестая версия, способная создавать межплатформенные приложения под управлением ОС Windows и Linux. В Delphi используется язык Object Pascal — объектно-ориентированная модификация языка Паскаль. Основу языка составляют классы, которые активно используют динамическую память, т. е. в этом языке устранен один из недостатков языка Паскаль, и можно работать с динамическими массивами переменной длины.

Delphi — мощная система визуального программирования, которая дает возможность разрабатывать программы любой сложности и характера. Инструменты интегрированной среды (см. рис. 16) позволяют значительно ускорить создание и отладку программ. Для многих типовых процедур программный код генерируется автомати-

чески. Путем выбора из списка достаточно легко создать управляющие элементы интерфейса. Широко применяется Delphi при программировании БД, приложений Windows, двух- и трехмерных графических приложений. С помощью версии Delphi 1 были разработаны приложения Windows, включающие оптимизирующий компилятор, визуальную среду программирования, мощные возможности работы с БД. С помощью версии Delphi 2 были разработаны аналогичные приложения для Windows 95 и Windows NT, а также большие библиотеки объектов. Версия Delphi 3 имеет расширенный набор инструментов для создания приложений, возможность использования технологии COM для разработки приложений WWW и др. В настоящее время появилась еще более совершенная версия Delphi 4 и 5. Подробнее о системе см. в теме 3.5.

Язык программирования *Ci* (C) разработан в 1972 г. Деннисом Ритчи и Б. Керниганом в фирме «Bell Laboratories» на основе языка *B*. Язык создавался для реализации ОС UNIX на ЭВМ PDP-11 фирмы DEC. *Ci* представлял собой дальнейшее развитие потомков Алгола 60 — языков *CPL* (разработан группой программистов из Кембриджского и Лондонского университетов, 1963 г.), *BCPL* (Basic Combined Programming Language, 1967 г., Мартин Ричардс, Кембридж) и *B* (Би) (Кен Томпсон, Bell Laboratories, г. Мюррей-Хилл, штат Нью-Джерси, 1970 г.).

С 1977 г. разрабатывались машинно-независимые трансляторы с языка *Ci*. В итоге этой работы появились совместимые по входному языку трансляторы для 15 различных типов ЭВМ. Сам язык тоже совершенствовался. В нем появились средства определения новых типов данных, широкий набор операторов, модульность и структурность, и в то же время в *Ci* включены средства программирования почти на уровне ассемблера (например, побитовые операции и работа с указателями). Язык позволяет избежать зависимости программ от конкретной ЭВМ и контролировать каждое действие алгоритма. *Ci* предоставляет обширные возможности при работе с памятью. В языке явно отсутствуют средства ввода-вывода, работы с файлами, работы со строками и другие, т. к. их использование обеспечивается вызовом соответствующих функций, что считается недостатком, хотя за счет этого экономится память, уменьшается размер компилятора, увеличивается быстродействие программ. Важное свойство языка — переносимость его программ на любые аппаратные платформы, что явилось причиной популярности *Ci* у системных программистов. Язык эффективен при создании компиляторов, драйверов, операционных систем. Он стандартизирован в 1989 г. и известен под именем ANSI C.

Язык *Ci* (в ходе работ по созданию мобильного транслятора с него) был переработан и усовершенствован. В 1979 г. Бьерн Страуструп — сотрудник «Bell Laboratories» научно-исследовательского центра AT&T (American Telephone and Telegraph) разработал язык «*Ci* с классами». За основу был взят язык *Ci* и добавлены элементы: из языка Simula-67 — конструкция *class*, а из Algol-68 — возможность перегрузки операций и свобода размещения объявления переменных. Проектирование, реализация и документирование новых возможностей языка происходили фактически одновременно. Язык был расширен средствами поддержки абстракций данных и объектно-ориентированного программирования. В 1983 г. язык стали называть *C++* (название дал Рик Массити, а операция инкремента ++, как известно, увеличивает значение переменной на единицу). В 1985 и 1990 гг. язык был модернизирован, в нем появились новые возможности: множественное и виртуальное наследование, шаблоны функций и классов, обработка исключительных ситуаций. В качестве методов (правил) обработки данных выступают функции, т. е. объект объединяет (инкапсулирует) в себе данные и функции, обрабатывающие эти данные. Объекты, имеющие одинаковые свойства, объединяют в группы, или классы, являющиеся центральным понятием этого языка. *C++* стал универсальным языком объектно-ориентированного программирования. В 1993 г. реализован коммерческий транслятор, который транслировал программу на *C++* и эквивалентную программу на *Ci*.

В 1990 г. язык был строго описан, и началась стандартизация языка и его библиотечных средств Объединенным комитетом ANSI-ISO (ANSI X3J16; ISO WG21/NO836). *C++* долгое время был открыт для расширений, что осложнило его стандартизацию. С момента начала стандартизации *C++* стал довольно громоздким языком, даже несколько изменилась его идеология. Сейчас окончательный вариант языка нуждается только в формальном одобрении, а практически новые компиляторы поддерживают все новые атрибуты *C++*. До сих пор язык считается эталоном объектно-ориентированного программирования.

Выпущены и визуальные системы программирования *C++* Builder фирмы «Borland» и Visual *C++* фирмы «Microsoft», основанные на языке *C++*, содержащие стандартный набор инструментов, характерных для визуальных систем: различные редакторы, возможность автоматической генерации программного кода, богатый набор утилит различного назначения (например для тестирования и отладки), средства программирования для Интернета. Система *C++* Builder пользуется большой популярностью у профессионалов.

Контрольные вопросы

1. Какова история развития языка программирования Pascal?
2. Что такое среда программирования Turbo Pascal?
3. Что такое система Delphi?
4. Какова история развития языка программирования C (Си)?
5. Что такое системы C++Builder и Visual C++?

2.3. Средства описания языков программирования. Основные понятия языков программирования

Средства описания языков программирования. Как указывалось, язык программирования должен иметь алфавит, синтаксис и семантику. Правила, определяющие структуру текста, составляют синтаксис (грамматику). Правила, позволяющие извлекать из текста смысл, называются семантикой языка.

Для описания синтаксиса языка используется так называемый *метаязык* (язык для описания языка). В качестве метаязыка используют либо металингвистические формулы Бэкуса—Наура, либо синтаксические диаграммы. Джон Бэкус разработал специальную систему определений для языков программирования. Например, определяя элемент «цифра», он указывал:

"<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |".

Этот способ записи называли нормальной формой Бэкуса (БНФ). Датский астроном П. Наур внес некоторые уточнения, и форму стали называть формой Бэкуса—Наура, но сокращенное название БНФ осталось.

В БНФ каждое понятие определяется с помощью одной металингвистической формулы, в левой части которой указываются определяемые понятия, а в правой — описываются все допустимые структуры языка, соответствующие этому понятию. Левая и правая части формулы связываются знаком $::=$, который читается «по определению есть». При перечислении дополнительных вариантов структур в правой части используется знак $|$, имеющий смысл «или». Все определяемые понятия языка программирования заключаются в угловые скобки $< >$, а неопределяемые исходные понятия языка называются *лексемами*. Лексемы могут использоваться в правой части формулы.

Например, определить понятие $<\text{целое число}>$, если алфавит языка программирования включает арабские цифры и знаки арифметических операций.

Сначала определим понятие $<\text{цифра}>$ перечислением с помощью метаформулы $<\text{цифра}> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |$.

Затем введем понятие $<\text{целое без знака}>$, описав его следующей формулой:

$<\text{целое без знака}> ::= <\text{цифра}> | <\text{целое без знака}> <\text{цифра}>$.

Это равносильно определению «любая последовательность цифр». Формула рекурсивна, т. к. $<\text{целое без знака}>$ уже встречается в правой части, т. е. это ссылка на само себя. Теперь выведем формулу:

$<\text{целое число}> ::= <\text{целое без знака}> | - <\text{целое без знака}> | + <\text{целое без знака}>$.

Полученная цепочка из трех формул определяет понятие $<\text{целое число}>$ через лексемы языка 0, 1, ..., 9, +, -.

С помощью металингвистических формул могут быть описаны все понятия языка программирования. Синтаксическая правильность текста проверяется в процессе его *синтаксического анализа* на основе металингвистических формул.

Способ описания синтаксиса на основе диаграмм основан на тех же принципах. Синтаксические диаграммы графически изображают допустимую структуру определяемого понятия. Следование элементов структуры задается стрелками, вариантность указывается как распараллеливание направлений следования. Лексемы изображают в виде кружка или овала, а определяемое понятие — прямоугольником. Диаграмма имеет одну входную и одну выходную стрелки.

На рис. 11 приводится изображение понятия «цифра» в виде диаграммы.

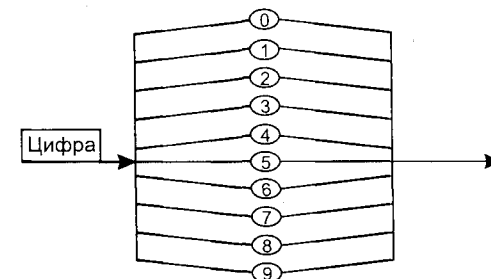


Рис. 11

Основные понятия языков программирования. Понятие *уровень языка* определяет уровень его абстракции. Язык низкого уровня соответствует реальному языку, который понимает процессор. Языки высокого уровня соответствуют не реальному, а некоторым гипотетическим процессорам.

Родственные конструкции разных языков различаются главным образом «внешним видом» — набором ключевых слов или порядком следования компонентив. Следовательно, конструкции современных языков имеют общую семантику (содержание), но различный порядок следования компонент (синтаксис) и разные ключевые слова (лексику).

Семантическое описание содержит, как правило, список компонент, из которых состоит конструкция (тип, имя типа), описание каждой компоненты, описание конструкции в целом. Синтаксическое описание дает формальное определение конструкции.

Большинство языков программирования оперируют одними и теми же *фундаментальными понятиями*, как, например, *тип данных* или *структура данных*.

Алгоритм содержит только указания на выполнение действий, а программа содержит еще некоторую дополнительную информацию. В большинство языков программирования включаются объекты двух видов: *описания* и *предписания*. В разделе *описаний* определяются объекты, с которыми манипулирует алгоритм (типы, структуры, имена и т. д.), задаются некоторые характеристики этих объектов. *Предписания* содержат указания на выполнение необходимых действий. Основным средством указания на выполнение некоторого действия в языках программирования является *оператор*. Операторы реализуют алгоритм решения задачи.

Как и в математике, в программировании принято оперировать *константами* и *переменными*. Есть простые переменные и переменные с индексами, которые используются для ссылки на некоторый элемент массива. Общим для большинства языков программирования является понятие *идентификатор*. Формально он определяется так:

$\langle \text{идентификатор} \rangle ::= \langle \text{буква} \rangle | \langle \text{идентификатор} \rangle \langle \text{буква} \rangle | \langle \text{идентификатор} \rangle \langle \text{цифра} \rangle,$

что означает: «идентификатор есть последовательность букв и цифр, начинающаяся с буквы». В основном идентификаторы употребляются в качестве *имен* каких-либо объектов (констант, переменных, типов, структур, программ и т. д.), что позволяет идентифицировать эти объекты.

Следующее понятие языков программирования — *выражение*. Оно бывает двух видов: арифметическое и логическое.

Понятие *арифметическое выражение* в точности соответствует принятому в математике. Пример: арифметическое выражение $(a + b)\sin x + c^2 \operatorname{tg}^5 y$ будет записано как $(a+b)*\sin x + c^2*\operatorname{tgy}^5$.

Логическое выражение образуется с помощью знаков логических операций из объектов (констант, переменных и т. д.) буквенного типа. В логическое выражение может входить *отношение* — еще одно из фундаментальных понятий языков программирования. Например, записи: $a < b, c >= d, x^2 + y^2 < a^2$ — отношения, а запись $(a < b) \text{ or } (c = n)$ — логическое выражение.

Важнейшим понятием является *подпрограмма* (понятие ввел американец Уилкс в 1957 г.). Понятие подпрограммы введено, чтобы избежать многократного повторения одинаковой последовательности предписаний. Подпрограмма может быть вызвана из любого места программы или другой подпрограммы с указанием тех значений исходных данных, с которыми она должна быть выполнена. Наиболее распространены: подпрограмма-функция и подпрограмма-процедура, которые различаются способом передачи результатов их выполнения (см. тему 1.2).

Контрольные вопросы

1. Что такое метаязык?
2. Что такое металингвистические формулы? Приведите пример.
3. Что такое синтаксические диаграммы? Приведите пример.
4. Каковы основные понятия языков программирования?
5. Что такое подпрограммы, для чего они предназначены?

Глава 3

ПРИНЦИПЫ СТРУКТУРНОГО ПРОГРАММИРОВАНИЯ. НАЧАЛА ПРОГРАММИРОВАНИЯ В СРЕДЕ TURBO PASCAL. СИСТЕМА DELPHI

В разделе даются вводные понятия о языке Pascal и основах программирования в средах и системах программирования, базой которых является этот язык.

3.1. Принципы структурного программирования.

Язык Pascal. Основы программирования в среде Turbo Pascal

Структурное программирование (см. также тему 1.1). Понятие связи и структур, образуемых ими, задач, которые могут быть решены на этих структурах, является основой структурного программирования.

Под *структурой* понимается определенный порядок объединения элементов, составляющих систему, или, структура — это множество связей между элементами системы. Поэтому структуру можно определить как множество (M) с определенными на нем отношениями (R). Если множество M конечно, а в отношении R вступает только пара элементов, то такую структуру изобразить просто, в виде *графа* (вершин, соединенных ребрами — сторонами).

Это понятие ввел великий математик Леонард Эйлер, рассматривая задачу о Кенигсбергских мостах: можно ли, гуляя по городу, так выбрать маршрут, чтобы пройти его один раз. В городе много островов, соединенных мостами. Из каждой вершины, кроме начальной и конечной, должно выходить четное количество ребер. Если вершина соединена сама с собой, то образуются петли. Условие направленности ребер отображается стрелкой, и тогда *граф* становится *ориентированным*. Такой граф несет гораздо больше информации, чем неориентированный.

Простая последовательная структура экземпляров (элементов) называется *очередь, линия*. При наличии разнообразных условий очередь выполнения действий может принимать разные виды структур: ветвления, цикла (многократного повторения). Для записи алгоритмических структур в виде блок-схем используется основное достоинство графа — наглядность записи структуры. А понятие «мыслить структурами» означает, что, исходя из условия задачи, нужно представить, какова будет структура алгоритма (следования, ветвления, цикла и т. п.) для дальнейшей записи его на языке программирования.

Идеи структурного программирования обосновал голландский ученый Дейкстра в 1969 г. Общим для языков структурного программирования (Паскаль, Ада, Модула-2, Си и др.) является удобство написания программ по правилам структурного программирования. Программы на этих языках строятся из четырех основных структур:

- последовательность действий;
- альтернативные (условные) действия;
- повторение действий (цикл);
- выделение вспомогательных алгоритмов.

Главным отличием этих языков от неструктурных языков (Фортран, Бейсик и др.) является ограничение или запрет на использование операторов перехода (GO TO), бездумное применение которых запутывает программу и затрудняет понимание ее логики.

Паскаль — язык структурного программирования для записи алгоритма решения задачи в виде программы для ЭВМ. Ниже приведены только основные сведения о языке. Для его глубокого изучения рекомендуется использовать специальную литературу.

При написании программы на этом языке сначала следуют ее заголовок, затем последовательно описать внешние модули, процедуры и функции, метки, константы, типы переменных и сами переменные, внутренние процедуры и функции и далее — операторы.

Для наименования различных величин и написания операторов в программах используется следующий *алфавит* языка:

- прописные буквы латинского алфавита от А до Z;
- прописные русские буквы от А до Я (используются только в строковых константах и пояснительных текстах — комментариях);
- арабские цифры от 0 до 9 (используются для записи чисел);
- знаки арифметических операций: + (сложение), — (вычитание), * (умножение), / (деление), DIV (деление нацело, с отбрасыванием остатка), MOD (нахождение остатка от деления нацело), ↑ или ^ (возведение в степень);

- знаки операций отношений: = (равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), <> (не равно);
- знаки логических операций: NOT (отрицание), OR (логическое сложение), AND (логическое умножение);
- знаки операций над множествами: * (пересечение множеств, \cap), + (объединение множеств, \cup), - (разность множеств), IN (принадлежность множеству);
- разделители и прочие символы: . (точка), , (запятая), ; (точка с запятой), : (двоеточие), _ (пробел — пустая позиция), () (скобки круглые), [] (скобки квадратные), { } (скобки фигурные), ' (апостроф) и др.

В языке зарезервировано примерно 80 слов, которые имеют определенное назначение и смысл, изменять их нельзя. Например, Program, unit, var, const, type, array и т. д.

Данные могут быть представлены как переменные и постоянные. Под переменной понимают содержимое ячейки (ячеек) оперативной памяти компьютера, которое может принимать разные значения. А задаваемым именем ячейки обозначают переменную.

Типы используемых данных (см. тему 1.2). Данные определяются числовыми, логическими и литерными (символьными) значениями и подразделяются на скалярные (имеют простейшую структуру) и структурированные (массивы, множества, записи и др.)

Типы скалярных данных:

INTEGER — целый,

REAL — действительный,

BOOLEAN — логический,

CHAR — символьный,

представляющие:

- числа целого типа (целые: от -32 768 до +32 768 для ЭВМ с 16 двоичными разрядами);
- числа действительного типа (имеют две формы записи: с фиксированной точкой — записываются в виде целой и дробной частей, разделенных десятичной точкой, например 3.10, -5.0 и т. п. и с плавающей точкой — используются для записи чисел в широком диапазоне значений, например: число 3 000 000,0 или $3,0 \cdot 10^6$ следует записать 3.0E6);
- данные логического (булевого) типа могут принимать значения: TRUE — истинно или FALSE — ложно;
- данные символьного типа (принимают значение символа, имеющегося в наборе литер данной ЭВМ).

Данные могут быть представлены перечисляемым типом (описание выполняется по схеме: TYPE <имя типа> = <список имен> — например, TYPE color=(white,red,blue);) или интервальным типом (задается интервалом значений, разделенных двумя точками, например: letter = 'A'...'Z';).

Для обозначения величин (констант, типов, переменных), а также функций, подпрограмм, файлов) используют имена — идентификаторы. Имя должно состоять из букв и цифр, но начинаться с буквы и иметь длину не более 8 символов (например: X, Y1, SUMMA, cross и т. д.). Эти имена задает пользователь. Но имеются и стандартные имена:

- констант: TRUE, FALSE, MAXINT;
- типов: INTEGER, REAL, BOOLEAN, CHAR, TEXT файлов: INPUT, OUTPUT;
- функций: ABS, SQR, SIN, COS, EXP, LN, ARCTAN, TRUNC, ROUND, PRED, SUCC, ORD, CHR, ODD, EOF, BOLN;
- процедур: GET, PUT, REWRITE, NEW, READ, UNPICK, PACK, READLN, RESET, WRITE, WRITELN, PAGE.

Простейшие конструкции языка:

- константы целого, действительного, логического, символьного и текстового типов (символьные и текстовые заключаются в апострофы или кавычки);
- переменные простые (записываются под своими именами, например: X1, b, Kul и т. п.) и с индексом (являются элементами массива, индекс задает местоположение элемента массива и записывается в квадратных скобках, например: A[1], B[4,5] — элемент на пересечении 1-й строки и 5-го столбца);
- стандартные функции, при использовании нужно записать имя, а в скобках указать аргумент (имена стандартных функций указаны выше); пример обращения: ROUND (X) — округление числа (смысл выполнения других стандартных функций см. в специальной литературе);
- выражения указывают правила выполнения операций над данными для получения требуемого результата; могут состоять из констант, переменных, стандартных функций, разделенных скобками и знаками операций, порядок выполнения их определяется скобками.

Операции выполняются слева направо и делятся на 4 группы:

- 1) отрицания: NOT;
- 2) умножения (мультипликативные): *, /, DIV, MOD, AND;
- 3) сложения (аддитивные): +, -, OR;
- 4) отношения: <=, <, =, < >, >, >=.

Правила записи выражений и выполнения операций:

- запись производится в одну строку без надстрочных и подстрочных знаков, например:

запись на языке Паскаль	математическая запись
$2.1 + A$	$2,1 + a$
$A^2 * B / C - D$	$a^2 \times b / c - d;$

- порядок выполнения операций: внутри скобок, вычисление встроенных функций, возведение в степень, умножение и деление, сложение и вычитание, при равном старшинстве — слева направо;
- два знака операций нельзя ставить рядом, а также опускать знак умножения между сомножителями.

Операторы.

BEGIN...END — составной оператор начала и конца процедуры, заключающий последовательность других операторов.

Оператор присваивания, имеющий вид:

<имя переменной> := <выражение>

Например: $c := 2 * p_i * r$; $X := 0.30$;

Операторы ввода:

READ ($x_1, x_2, \dots, x_i, \dots, x_n$) — обеспечивает выборку данных, из стандартного входного файла INPUT, где $x_1, x_2, \dots, x_i, \dots, x_n$ — список имен переменных.

READLN ($x_1, x_2, \dots, x_i, \dots, x_n$) — обеспечивает то же, что и предыдущий оператор, а после выборки выполняет переход к началу новой строки файла.

READLN — обеспечивает пропуск одной строки в файле INPUT и переход к началу новой строки.

Операторы вывода:

WRITE ($x_1, x_2, \dots, x_i, \dots, x_n$) — обеспечивает вывод значений, соответствующих именам переменных: $x_1, x_2, \dots, x_i, \dots, x_n$ в стандартный файл OUTPUT.

WRITELN ($x_1, x_2, \dots, x_i, \dots, x_n$) — обеспечивает то же, что и предыдущий оператор, а после выборки выполняет переход к началу новой строки файла.

WRITELN — обеспечивает пропуск одной строки в файле OUTPUT и переход к началу новой строки.

Структурные операторы — см. темы 3.2—3.3.

Операторы, реализующие ситуацию развилки:

1. Условный

IF <логическое выражение>
THEN <оператор1>
ELSE <оператор2>

2. Оператор варианта (выбора)

CASE <выражение> OF
<список констант1> : <оператор1>;
<список констант2> : <оператор2>;
.....
<список констант N> : <операторN>;
END

Операторы цикла:

FOR — организует цикл с известным числом повторений и имеет вид:

FOR $i = n_1$ TO n_2 DO s ,

где

i — параметр цикла целого типа;

n_1, n_2 — начальное и конечное значения параметра цикла;

s — тело цикла, содержащее один или несколько операторов.

Если оператор записать в виде:

FOR $i = n_2$ DOWNT0 n_1 DO s ,

то здесь ключевое слово DOWNT0 меняет шаг в цикле на обратный: -1 .

WHILE — организует цикл с неизвестным числом повторений, с предусловием, и имеет вид:

WHILE b DO s ,

где b — логическое выражение; s — тело цикла, содержащее один или несколько операторов.

REPEAT — организует цикл с неизвестным числом повторений, с постусловием, и имеет вид:

REPEAT s UNTIL b ,

где b — логическое выражение; s — тело цикла, содержащее один или несколько операторов.

Описание других операторов см. в специальной литературе.

Программа состоит из заголовка, раздела описаний и блока операторов, завершающегося точкой. Каждый раздел начинается со стандартного служебного слова. Заголовок пишется в первой строке и начинается с ключевого слова **PROGRAM**, а затем следуют имя

программы и перечисляются файлы, доступные ей. Например, Program Dva или Program Summa (INPUT, OUTPUT), где **INPUT**, **OUTPUT** — стандартные файлы ввода и вывода, которые используются для связи операторов программы с исходными данными и результатами их обработки.

Каждое описание и определение заканчивается *точкой с запятой*. Метки (**LABEL**) задаются операторам, которым передается управление. *Метки* — целые числа, содержащие не более четырех цифр, например: **LABEL 3, 125**. От оператора они отделяются двоеточием. Если меток нет, то раздел их описания в программе отсутствует.

Константы описывают с помощью ключевого слова **CONST**. Имя и значение константы разделены символом **=**. Например **CONST PI=3.1416**;

Переменные описывают с помощью ключевого слова **VAR** и указанием списка переменных и их типов в следующем виде: **VAR (x₁, x₂, x₃, ..., x_n):T**; . Например:

```
var
A, C: real;
K, L: integer;
M: char;
S: boolean;
```

Массив описывается так: **A: ARRAY [K...L] OF T**;

Например: **A: ARRAY[1...20] OF REAL**;

Процедуры и функции описываются под заголовками **PROCEDURE** и **FUNCTION**.

При выводе значений необходимое число пробелов задают указанием пробела в апострофах и константой целого типа. Например: **WRITELN (' ': 5, 'Константы');** — печатается 5 пробелов, а потом — слово 'Константы'.

Размер поля (количество позиций) также задают в явном виде. Например: **WRITELN ('A':7:5)**, т. е. для значения A выделяется 7 позиций, 5 из них — дробная часть.

Пример. Запись простейшей программы:

```
Program Dva;
var
a, b: integer;
begin
  writeline ('Введите, пожалуйста, два равных числа с пробелом
            между ними');
  read (a, b);
  if a = b then
```

```
writeline ('Спасибо');
else
  writeline ('Два числа должны быть равны !');
end.
```

Теперь рассмотрим начала программирования в среде Турбо Паскаль.

Для запуска системы нужно щелкнуть мышью на ярлыке BPascal на рабочем столе (B — аббревиатура фирмы Borland) или набрать команду, например: **D:\TP\TURBO**.

Изображение основного окна программной оболочки см. на рис. 12. Интегрированная среда программирования включает: компилятор, редактор связей, средства отладки, обеспечивающие полную проверку переменных, данных и выражений. Основные сведения о среде см. в теме 1.4.

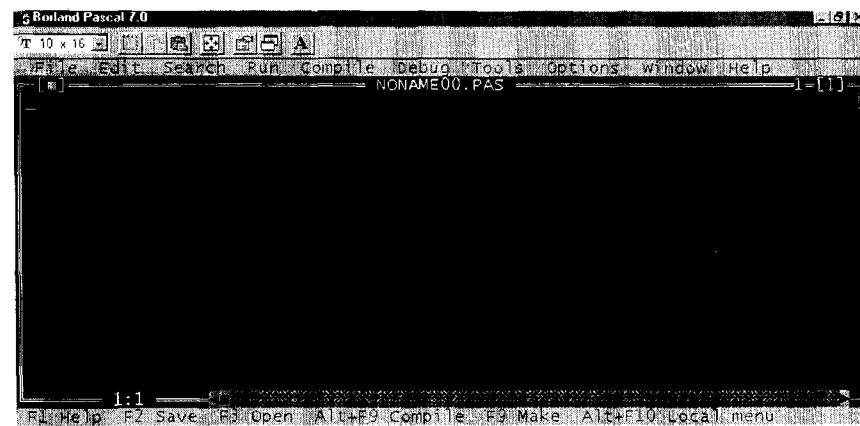


Рис. 12

Рабочие инструменты окна, в котором выполняется программирование, имеют следующие элементы: все пункты главного меню; раскрывающиеся подменю; пиктограммы; меню в нижней части окна. Элементы диалоговой среды: меню, окна, поля.

Для управления средой программирования используются следующие *функциональные клавиши* в верхнем ряду клавиатуры:

F1 — открывает экран справочной информации;

F2 — сохраняет файл, созданный в активном окне редактора;

F3 — открывает файл;

F4 — выполняет фрагмент программы от подсвеченной строки до курсора;

F5 — раскрывает активное окно на полный экран или возвращает его к исходному размеру;

F6 — переход к следующему открытому окну (последовательная смена окон);

F7 — выполняет одну строку программы с заходом внутрь процедур;

F8 — выполняет одну строку программы без вызова процедур;

F9 — компилирует программу;

F10 — переход к выбору из главного меню.

Действия этих клавиш можно заменить комбинациями с клавишами **Alt** (от *Alternative* — дополнительный), **Ctrl** (от *Control* — управление) и **Shift** (сдвиг). Далее приводятся назначение нескольких комбинаций клавиш:

Alt+F9 — выполнить прогон программы, компилировать программу, находящуюся в редакторе, загрузить ее в оперативную память и выполнить, после чего вернуться в среду программирования;

Alt+F5 — сменить окно редактора на окно вывода результатов работы программы;

Ctrl+F5 — изменить положение и размеры активного окна;

Ctrl+F9 — прогон программы;

Shift+F1 — выбор справки из списка сообщений;

Ctrl+F1 — справка о нужной стандартной процедуре;

Alt+F1 — получение предыдущей справки.

Создание программ в среде Турбо Паскаль осуществляется в окне текстового редактора среды программирования в соответствии с разработанным алгоритмом. В окне создается, открывается, редактируется текст программы, она запускается на выполнение.

Составление программы осуществляют набором констант, переменных, операторов и т. д., как правило, указывая их в определенной алгоритмом последовательности. Значимые минимальные единицы текста в программе на языке Паскаль представляются специальными символами, зарезервированными словами, идентификаторами, метками, числами и строковыми константами.

Пример. Записать в окне редактора текст программы, в результате выполнения которой, вы хотите получить частное от деления двух вводимых чисел.

```
Program Delenie
var
  x1, x2, y: real;
begin
  writeln ('Введите число x1');
```

```
  readln (x1);
  writeln ('Введите число x2');
  readln (x2);
  y:=x1/x2;
  writeln ('Результат деления x1/x2=', y);
end.
```

В конце каждой строки редактор вставляет невидимый для глаза на экране символ-разделитель. Этот символ вставляется «вводом» и стирается «забоем» или **Del**. С помощью вставки и стирания можно разрезать и склеить строки.

Нормальный режим работы — вставка. Смена режимов — **Ins** (вставка).

Написав текст программы, его следует сохранить в файле с оригинальным именем и с помощью команды **Run** запустить на выполнение. При обнаружении ошибок следует выполнить отладку программы, используя команды меню **Debug**.

Контрольные вопросы

1. Что такое структурное программирование?
2. Какие типы данных используются в языке Паскаль?
3. Какие операторы языка Паскаль вы знаете, назовите их назначение?
4. Каковы основные конструкции языка Паскаль?
5. Каковы основные составляющие среды программирования Турбо Паскаль?

3.2. Программирование линейных задач в среде Turbo Pascal

В программе линейной структуры все действия записываются строго последовательно, одно за другим в соответствии с разработанным линейным алгоритмом.

Составление программы линейной структуры осуществляют описанием переменных, констант, а также набором последовательно записанных команд.

В теме 1.2 указывались этапы разработки программы. Рассмотрим примеры создания *программ линейной структуры* в соответствии со следующими этапами:

- постановка задачи;
- определение метода решения;

- составление сценария работы с ЭВМ;
- разработка алгоритма;
- перевод алгоритма в программу;
- ввод и тестирование программы;
- анализ полученных результатов.

Задача.

1. Постановка (условие) задачи.

Вычислить объем пирамиды, основанием которой является треугольник.

2. Метод решения.

Для вычисления использовать формулу Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)} \text{ — площадь основания,}$$

где $p = (a + b + c)/2$.

Объем пирамиды $V = 1/3sh$, где h — высота.

Программирование следует начинать со сценария — определения типа переменных, их разрядности, порядка ввода, вывода результата.

3. Сценарий.

а) по формуле Герона составим пропорцию для вычисления площади треугольника через его стороны (a, b, c) и полусумму $p = (a + b + c)/2$. Для нахождения полусуммы сторон использовать переменную p , площади основания — переменную S , объема пирамиды — переменную V . Для сторон использовать имена переменных a, b, c , высоты — h . Все переменные вещественного типа;

- записать алгоритм решения задачи в виде блок-схемы;
- результат решения задачи вывести на экран.

4. Алгоритм (рис. 13).

5. Программа.

```

Program Piramida (INPUT, OUTPUT);
var : a, b, c, h, p, S, V: real;
begin
  read (a, b, c, h);
  p:=(a+b+c)/2;
  S:=sqrt (p*(p-a)*(p-b)*(p-c));
  V:=S*h/3;
  writeln ('Объем пирамиды V=', V)
end.
```

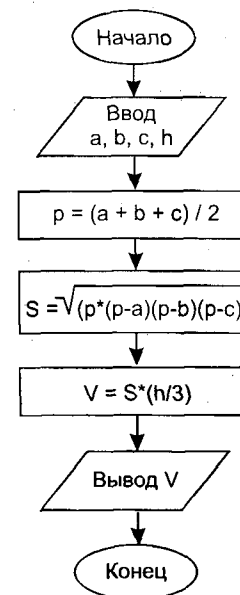


Рис. 13

6. Результат выполнения программы.

Запустить программу на выполнение с помощью системной команды RUN. Исправить ошибки, т. е. выполнить отладку программы.

Ввести значения сторон и высоты треугольника: a, b, c, h .

Ввод: четырехзначные числа (2 знака после запятой).

Ответ на экране: Объем пирамиды $V=$ (четырезначное число, 2 знака после запятой).

Особый интерес представляет работа с графикой.

Задача.

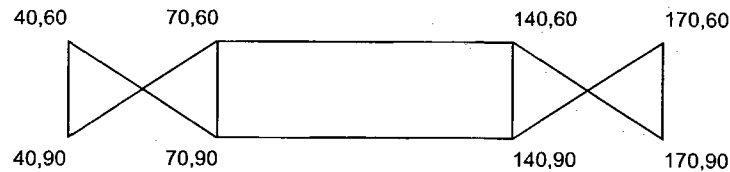
1. Постановка (условие) задачи.

Составить программу рисования конфеты.

2. Метод решения.

Программирование следует начинать со сценария — рисунка на бумаге в клеточку. Рисунок следует снабдить системой координат, расположенной так же, как на экране ЭВМ, задавая координаты x, y .

3. Сценарий рисунка.



Записать алгоритм рисования картинki.

4. Алгоритм изображения конфеты.

АЛГ Конфета

Нач

прямоугольник (70,60 — 140,90)

линия (40,90 — 70,60)

линия (40,60 — 70,90)

линия (140,60 — 170,90)

линия (140,90 — 170,60)

линия (40,60 — 40,90)

линия (170,60 — 170,90)

Кон

5. Программа рисования конфеты.

PROGRAM KONFETA;

BEGIN

Uses Graph; {подключение графического модуля}

SETLINESTYLE (4,4,1); {4 — стиль линии, 4 — образец стиля}

{1 — толщина линии нормальная}

RECTANGLE(70,60, 140,90); {рисование прямоугольника}

LINE(40,90, 70,60); {рисование линии}

LINE (40,60, 70,90);

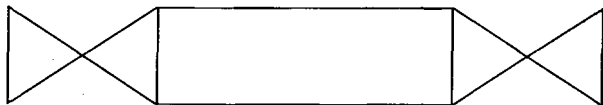
LINE (140,90, 170,60);

LINE (40,60, 40,90);

LINE (170,60, 170,90)

END.

6. Результат выполнения программы: рисунок на экране.



Контрольные вопросы

1. Каковы этапы решения задач на ЭВМ?
2. Что такое программа линейной структуры?
3. Что такое метод решения задачи?
4. Что такое сценарий?
5. Что такое отладка программы?

3.3. Программирование задач разветвляющейся структуры в среде Turbo Pascal

Составление программы разветвляющейся структуры осуществляют с помощью операторов условий и выбора. Действия в программе записываются в зависимости от выполнения (невыполнения) некоторого условия в соответствии с разработанным алгоритмом ветвления.

В структуру ветвления входят служебные слова: *если*, *то*, *иначе*, и она может иметь вид 1, 2 или 3.

1) *если* условие (логическое выражение)

то действие 1

все

2) *если* условие (логическое выражение)

то действие 1 (один и более операторов)

иначе действие 2 (один и более операторов)

все (означает конец структуры)

3) Для сложных ветвлений эти операторы могут быть *вложенными*, т. е. иметь следующую структуру:

если условие 1

то действие 1

иначе

если условие 2

то действие 2

.....

если условие n

то действие n

все (означает конец структуры)

1) *if* условие

then оператор

end;

2) *if* условие

then оператор 1

else оператор 2

end;

3) *if* условие 1

then оператор 1

else

if условие 2

then оператор 2

.....

if условие n

then оператор n

end;

При количестве условий больше двух удобно пользоваться структурой *выбор*, которая имеет следующий вид:

<i>выбор из</i>	case выражение of
список констант l: действие l	список констант l: оператор l;
.....
список констант n: действие n	список констант n: оператор n;
<i>все</i>	end;

Пример программы с оператором выбора см. в теме 1.3.

Создание программ на языке осуществляется в окне редактора среды программирования в соответствии с разработанным алгоритмом.

Далее приведен пример создания *программы с условиями* в соответствии с вышеуказанными этапами.

Задача. Найти большее из двух чисел.

1. *Постановка задачи.*

Нахождение большего из двух чисел.

2. *Метод решения.*

Программирование следует начинать со сценария — определения числа вариантов решения задачи, типа переменных, их разрядности, порядка ввода, вывода результата.

3. *Сценарий.*

а) наметить два варианта нахождения большего из вещественных чисел с двумя десятичными знаками;

б) записать алгоритмы нахождения большего из двух чисел на алгоритмическом учебном языке;

в) написать программу. Выполнить и вывести результат решения задачи на ЭВМ.

4. *Алгоритм.*

АЛГ Первый способ (арг. вещ. a,b, рез. вещ.)

НАЧ

ввод ('Введите первое число:', a)

ввод (('Введите второе число:', b)

если a>b

то b:=a

вывод ('большее', b)

все

КОН

АЛГ Второй способ (арг. вещ. a,b, рез. вещ. m)

НАЧ

вывод ('два числа')

ввод (a,b)

если a>b

то m:=a

иначе (m:=b)

все

вывод ("большее", m)

КОН

5. *Программа (для второго способа).*

Program Vtorspos;

var

a, b, m: real;

begin

writeline ('Два числа');

read (a);

read (b);

if a>b then

m:=a

else

m:=b

writeline ('Большее =', m);

end;

end.

Контрольные вопросы

1. Что такое программа разветвляющейся структуры?
2. Каковы функции оператора **if ... then**?
3. Когда используется оператора **if ... then ... else**?
4. Когда используются вложенные операторы?
5. Каково назначение оператора **case**?

3.4. Программирование задач циклической структуры в среде Turbo Pascal

На практике часто возникает необходимость повторить определенные действия несколько раз. Последовательность действий многократно повторяющихся называется *циклом*. Запись алгоритмов *циклической структуры* приведена на рис. 6.

Составление программы циклической структуры осуществляют с помощью команд языка программирования в окне редактора среды программирования в соответствии с разработанным алгоритмом.

Для представления циклов в Pascal-программах имеются три типа стандартных операторов: цикл с параметром, цикл с предусловием и цикл с постусловием.

Циклы с параметром используются, если какие-либо действия нужно выполнить определенное число раз. Между началом и концом цикла заключаются действия, выполняющиеся нужное число раз. После служебного слова «для» стоит параметр (счетчик) цикла с указанием его начального и конечного значения и шагом изменения.

Форма записи цикла с параметром следующая:

```
for параметр := начало to конец do оператор (при возрастании
параметра с шагом 1);
for параметр := начало downto конец do оператор (при убывании
параметра с шагом -1),
```

где:

параметр — числовая переменная;
начало — начальное значение параметра;
конец — конечное значение параметра;
шаг — величина изменения значения *счетчика*.

Ниже приведен пример создания алгоритма и программы, имеющей цикл с параметром.

Задача. Вывести на экран 5 раз ваше имя.

```
АЛГ
НАЧ цел k 'k — параметр цикла
нц для k от 1 до 5
    вывод ("Имя")
кц
КОН
```

```
Program Imja;
const a:='Александр';
var k:=integer;
begin
    for k:=1 to 5 do
        writeln ('Мое имя', a:9);
    end;
end.
```

Рассмотрим пример поэтапного создания программы циклической структуры.

Задача.

1. *Постановка задачи.*

Ввести значения четырех чисел и найти их сумму.

2. *Метод решения.*

Программирование следует начинать со сценария — определения варианта решения задачи, типа переменных, вида цикла, порядка ввода, вывода результата.

3. *Сценарий.*

а) для нахождения суммы использовать переменную *s*, начальное значение которой равно 0. Для введенного числа использовать переменную *a*. Переменной *i* обозначить параметр (счетчик) цикла.

б) записать алгоритм нахождения суммы четырех чисел на учебном алгоритмическом языке;

в) написать программу. Выполнить и вывести результат решения задачи на ЭВМ.

4. *Алгоритм.*

```
АЛГ Сумма чисел (арг. цел. a, рез. цел. s)
НАЧ цел i
s:=0
нц для i от 1 до 4
    вывод ('Ввести число')
    ввод (a)
    s:= s+a
кц
    вывод (s)
КОН
```

5. *Программа нахождения суммы четырех чисел.*

```
Program S4;
var
a, i, s: integer;
begin
    s := 0; {объявление начального значения переменной}
    writeline ('Введите числа');
    for i:=1 to 4 {начало цикла с шагом =1}
        readln (a);
        s := s+a;
        writeln ('Сумма этих чисел =', s)
    end;
end.
```

Результат выполнения программы, будет, например, следующий:

Введите любое число: 10
 Введите любое число: 5
 Введите любое число: 20
 Введите любое число: 7
 Сумма этих чисел = 42

Для обозначения циклов по условию используются цикл с предусловием и цикл с постусловием.

Цикл с предусловием — это повторение действий с предварительной проверкой. Во многих задачах следует вначале проверить, можно ли выполнить цикл даже в первый раз.

Цикл с предусловием имеет следующий формат записи:

while условие (логическое выражение) **do** операторы

На алгоритмическом языке эта структура имеет вид:

нц 'начало цикла
 пока 'запись условия продолжения цикла
 тело цикла 'операторы
 кц 'конец цикла

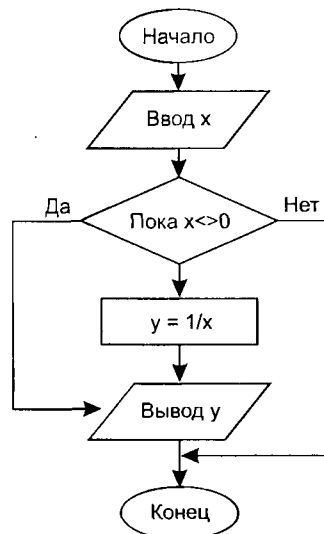


Рис. 14

Последовательность действий, заключенных между началом и концом цикла, называется *телом цикла* и выполняется, если условие цикла истинно.

Приведем пример алгоритма и программы, имеющей цикл с предусловием (рис. 14).

Задача. Вычислять обратную величину введенного по запросу числа до тех пор, пока не будет введен 0, после этого должен быть выдан результат.

Program Obratn;

var

x, y: real;

begin

writeln ('Введите произвольное целое число');

read (x);

while x<>0 do; {0 как первое число вводить нельзя,
 {т. к. никакое число нельзя
 {делить на 0}

begin

y:=1/x;

writeln ('Обратная величина =', y:4:2);

writeln ('Введите произвольное целое число');

read (x);

end;

writeln ('На ноль делить нельзя');

end.

Цикл с постусловием — это многократное повторение действий (операторов) до тех пор, пока не будет введено условие.

Цикл с постусловием имеет следующий формат записи:

repeat последовательность операторов **until** логическое выражение

На алгоритмическом языке эта структура имеет вид:

нц 'начало цикла
 повторять до тех пор 'запись условия продолжения цикла
 тело цикла 'операторы
 пока
 кц 'конец цикла

Приведем пример алгоритма и программы, имеющей цикл с постусловием (рис. 15).

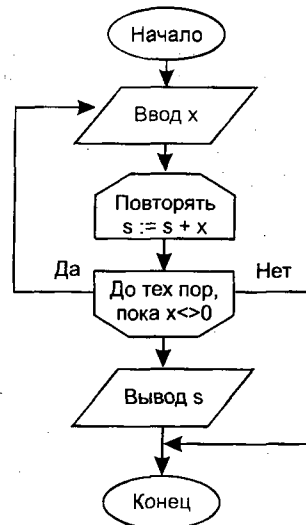


Рис. 15

Задача. Запрашивать целые числа, их суммировать и выводить суммы на экран до тех пор, пока не будет введен 0.

```

Program S_post;
var
  x, s: integer;
begin
  s := 0; {объявление начального значения переменной}
  repeat
    writeline ('Введите числа для суммирования (если
               ввести 0 — конец)');
    read (x);
    s := s+x;
  until x=0;
  writeln ('Сумма =', s)
end.
  
```

Рассмотрим еще пример поэтапного создания программы циклической структуры.

Задача. Вводить имена персонажей сказки «Репка» до тех пор, пока они не вытащат Репку.

1. Постановка задачи.

Ввести значения действий и найти сумму действий, через которые вытащат Репку.

2. Метод решения.

Программирование следует начинать со сценария — определения варианта решения задачи, типа переменных, вида цикла, порядка ввода, вывода результата.

3. Сценарий.

а) для нахождения суммы использовать переменную s , начальное значение которой равно 0. Для наименования действия использовать переменную a . Переменной k обозначить параметр (счетчик) цикла;

б) записать алгоритм нахождения суммы действий персонажей;

в) написать программу. Выполнить и вывести результат решения задачи на ЭВМ.

4. Алгоритм.

АЛГ Репка (аргцел, резцел)

НАЧ цел k ; симв s , a

$s := 0$

нц для k от 1 до 5

 вывод ("Ввести текст")

 пока $a < 5$

 вывод ('КОГО?');

 ввод (a)

$s := s + a$

 если $s \leq 5$ то $k := k + 1$

 вывод ('Тянут-потянут, вытянуть не могут');

 иначе

 вывод (s)

 вывод ('Тянут-потянут, вытянули Репку');

кц

КОН

5. Программа Репка.

Program Repka;

var

k : integer;

s , a , b : char;

writeln ('Посадил дед репку. Выросла репка большая, пребольшая.');

writeln (' Дед тянет-потянет — вытащить не может. Позвал...');

begin

 repeat

 writeln ('КТО?');

 readln (b);

 writeln ('КОГО?');

```

readln (a);
writeln ('Тянут-потянут, вытянуть не могут');
until a="мышку";
writeln ('Тянут-потянут, вытянули Репку');
end;
end.

```

Контрольные вопросы

1. Что такое программа циклической структуры?
2. Каковы функции оператора `for ... to ... do`?
3. Каковы функции оператора `for ... downto ... do`?
4. Что такое цикл с предусловием и какова форма его записи?
5. Что такое цикл с постусловием и какова форма его записи?

3.5. Delphi — система программирования задач на языке Object Pascal

Delphi — одна из популярных современных систем для разработки программ, т. е. автоматизированная система программирования, работающая на платформе ОС Windows. Delphi разработана на фирме «Borland» исследовательской группой под руководством Чака Язджевски. Название системы придумал Д. Торп, дав ей имя древнегреческого города Дельфы, где находилось святилище бога мудрости и покровителя искусств Аполлона и жриц, которые изрекали пророчества. Delphi представляет собой мощное инструментальное визуальное средство, позволяющее в наглядной форме моделировать предметную область, анализировать эту модель и разрабатывать программные приложения в соответствии с информационными потребностями пользователя. В среде системы разработаны многочисленные приложения в различных сферах науки, производства и т. д.

Программирование в среде Delphi осуществляется на языке объектного программирования Object Pascal (см. главу «История развития программирования»). Разработал его Н. Вирт совместно с группой специалистов фирмы «Apple». Язык является объектно-ориентированным расширением языка Паскаль. Классы (типы) реализованы как расширение структуры RECORD и могут содержать как поля данных, так и поля методов. Методы такие же, как процедуры или функции Паскаля, определяемые по имени класса. Сообщения посылаются с помощью обычных Pascal-конструкций

для определения поля. Библиотека классов Object Pascal составляет основу многих интегрированных систем для разработки прикладных программ.

Проекты в Delphi состоят из файлов. Многие из этих файлов автоматически генерирует сама система. Внешний вид (интерфейс) создается из набора стандартных элементов. Программисту, а чаще просвещенному пользователю, остается только написать программу действий имеющихся элементов. В основе Delphi — технология, получившая название Two Ways Tools, т. е. при размещении или изменении компонента в форме соответствующий программный код будет автоматически дополнен или изменен, а при изменении программного кода компонент в форме меняется.

Если Turbo Pascal работает в экранном текстовом режиме ОС DOS, то Delphi — в графическом режиме, как ОС Windows. Однако большинство программ, созданных в среде Turbo Pascal, можно адаптировать в среду Delphi. Компоненты (объекты) готовых программ Delphi очень эффективно и удобно можно использовать для создания визуальных оболочек проектов.

Delphi поддерживает все основные приемы работы в ОС Windows и имеет такую же структуру хранения программ и данных. Для каждого проекта целесообразно создать свою отдельную папку, а все ваши проекты собрать в одну общую. Например, D:\Pap\Project1, D:\Pap\Project2, D:\Pap\Project3.

Вызвав программу Delphi32.exe, можно увидеть внешний вид инструментальной среды (рис. 16).

В основном окне имеются: главное меню, панель инструментов (горячих кнопок), панель и палитры компонентов; окно инспектора объектов; окно формы проекта и окно модуля программирования.

В самой верхней строке экрана дан заголовок Delphi — Project1, где отображаются заголовки проектов.

Всплывающие на экране надписи-подсказки позволяют быстро ознакомиться с:

- функциями главного меню среды Delphi, в том числе с ImageEditor (редактор изображений) из подменю Tools главного меню; этот инструмент похож на графический редактор Paint, имеющийся среди стандартных программ;
- работой горячих кнопок (инструментов), расположенных в два ряда, в виде пиктограмм слева под главным меню;
- панелью вкладок и палитрой их компонент, расположенных справа под главным меню;
- свойствами (Properties) и событиями (Events) Инспектора объектов (Object Inspector);

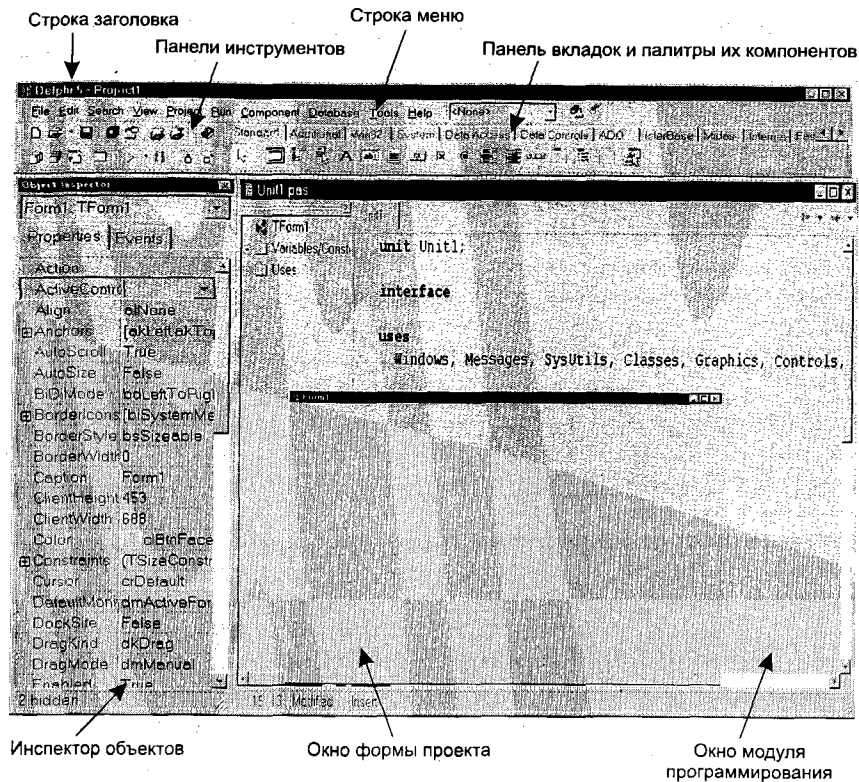


Рис. 16

- формой проекта (*Form1*) и модулем программирования (*Unit1.pas*).

Горячие кнопки — краткая форма функций меню.

Панель компонент содержит вкладки, при активизации которых открывается панель значков с указанием их значений — готовых деталей для конструирования проекта.

В левой части экрана находится окно инспектора объектов — *Object Inspector*. Иногда его называют паспорт или досье. В левой части находятся названия полей, а в правой отображаются свойства объектов и события. Свойства отображают информацию о текущем объекте в окне формы. Можно изменять эти свойства. События определяют действия, которые объект должен выполнять, если происходят определенные события с объектом. В Инспекторе объектов задаются и исправляются важные характеристики объекта, имя ко-

торого находится в заголовке. Например, найдя в Инспекторе объектов соответствующие поля, можно задать в поле *Caption* (Заголовок) вместо *Form1* — свой заголовок проекта, в поле *Width* — ширину формы, в поле *Height* — высоту формы, в поле *Font* — шрифт и т. д. Но заголовок формы и ее имя не одно и то же. Чтобы заголовок остался, его нужно задать также в поле *Name* (Имя).

Окно формы проекта может занимать большую часть экрана, размеры окон можно менять. В форме (первоначальное название — *Form1*) конструируют проекты. Все элементы управления размещают в этом окне, перемещая и меняя их размеры. То, что проектируют в форме, очень похоже на то, что получают в результате выполнения программы. Реальная программа может содержать несколько форм.

Чтобы показать нужную форму на экране, следует либо выбрать в меню команду *View* (Просмотр) и далее в открывшемся списке форм найти нужный заголовок, либо на панели быстрых кнопок выбрать пиктограмму просмотра форм (*View Form*).

Окно модуля программирования Unit1 получают, закрыв окно формы, а затем восстановив его. Это окно всегда находилось на экране, но было не активно, как бы «спрятано» за окном формы. Окно модуля содержит редактор, в котором записывают исходный текст программы для Delphi-приложения. Первоначально оно содержит одну страницу, хранящую исходный текст для новой формы, которую собираются создать. Код интерфейса пользователя в основном генерируется в Delphi автоматически, а не пишется вручную. Пользователь пишет только код «склейки», который собирает компоненты вместе: 2–3 строки, определяющие, какие действия следует выполнять при появлении конкретного события.

Как правило, постоянно работают то с окном формы, то с окном модуля. Переключать одно на другое можно быстро с помощью клавиши *F12*.

В системе есть уже автоматически созданный модуль исполнения программ. Поэтому, нажав клавишу *F9*, можно увидеть раскрывшееся окно формы с ярлыком в виде горящего факела: факел горит — программа выполняется, если факел погашен — исполнение программы прекращается (с помощью кнопки-перекрестия в правом верхнем углу формы). Изменения в программу следует вносить, если факел погашен, т. е. на стадии проектирования, а не на стадии исполнения программы, когда факел горит.

На стадии проектирования необходимо хорошо осмыслить назначение проекта и грамотно подобрать все компоненты и параметры. На вводимые события программа должна реагировать определенным образом.

Delphi — диалоговая система, т. е. способная выполнять оперативный обмен сообщениями между пользователем и ЭВМ. В системе диалог может быть организован как *примитивный*, т. е. собеседники не очень обращают внимание на ответы друг друга; *одноступенчатый*, т. е. один вопрос — один ответ; *двухступенчатый*, т. е. сначала следует один вопрос, а после ответа на него — второй, в зависимости от ответа на первый; *многоступенчатый*, аналогичен двухступенчатому, но с разным количеством вопросов и ответов. Чаще всего используют двухступенчатый диалог. Диалог может строиться на выборе фраз из массива. В Delphi можно программировать анимационные проекты, писать программы, предназначенные для сети Internet, организовывать ссылки на различные сайты Всемирной паутины (см. тему 4.2).

Самый простой способ дать указание программе о событии — это выбрать с помощью мыши на Панели компонент определенную вкладку, содержащую в нижнем ряду цветные значки компонент, — готовые детали для конструирования проекта. Например, вкладки: Standard — стандартные компоненты, Additional — дополнительные компоненты, System — системные компоненты и т. д.

При программировании в Delphi нужно знать, что базовые конструкции языка Object Pascal уже имеются в среде и вызываются автоматически.

Принципиально важное соглашение языка ObjectPascal: каждый объект, используемый в программе, должен быть предварительно описан. Часть из них описывает сама система. Перед началом процедуры (перед *begin*) следует описать (после служебного слова *var*) все придуманные переменные, с которыми система не знакома. Используют локальные и глобальные имена переменных. Глобальные имена описывают, помещая *var* в верхней части большого раздела Implementation (исполнение), в котором создаются заготовки всех процедур.

Запись (*record*) может объединять в себе описания различных типов объектов. Служебные слова *record...end* выделяют блок записи, где описывают имена и типы полей записи.

В Object Pascal объект является структурой данных. В объекте код и данные представляют собой единый логический пакет. Класс — это тип объекта. Определение класса (CLASS) используется для определения экземпляра класса, что наиболее правильно называть объектом. При описании типа объекта после CLASS стоит указание на класс объекта, уже существующего в Delphi. Например: TPredmeti = CLASS(TObject).

Тип множества задает сам программист. Для этого перед разделом описания переменных помещают еще один раздел с именем типа (тип), предварив имя буквой *T*. Например:

```
type
TSet of Data = set of 1,...,26;    // тип — диапазон
TSet of Chars = set of 'A',..., 'Z';
```

Имеются и динамические структуры данных: указатели и динамические списки. Если нужно хранить в оперативной памяти величину типа *integer*, то, описав *var P: ^integer*, знают, что *^* указывает системе, что в переменной *P* должен храниться адрес величины типа *integer*.

Программирование на языке Object Pascal основано на знании логических средств, процедур и функций, средств организации циклов.

Используются такие основные логические средства языка, как конструкция *if... then... else...*; знаки операций сравнения *<*, *>*, *=*, *<=*, *>=*, *<>*; специальные логические операции *or* (или), *xor* (исключающее или), *and* (и), *not* (не). Эти операции имеют следующий приоритет: *not*, *and*, *or* и *xor*, *=*, *<*, *>*, *<>*, *<=*, *>=*.

Например, в задаче: увеличить ширину кнопки на 20, если она меньше 90, а если больше, уменьшить ее в 3 раза, конструкцию в программном модуле следует записать в следующем виде:

```
if Button1. Width < 90 then
Button1. Width := Button1. Width + 20
else Button1. Width := Button1. Width div 3;
```

При программировании между служебными словами *begin* и *end* (их именуют еще операторными скобками) можно написать любой блок программы. Короткие комментарии отделяют двойными слэшами, длинные — фигурными скобками или звездочками.

Файлом в Object Pascal называют последовательность компонентов (данных) одного типа. Есть несколько разновидностей файлов, наиболее часто используемые — типизированные файлы. Для их описания используется следующая конструкция:

```
type
имя типа файла = file of тип компонентов файла;
```

Например:

```
type
TFile Numb: file of integer;
TFile Spisok Table: file of TSpisok Table; .
```

Файл в программе представляет файловая переменная (F), которая описывается следующим образом:

```
var
  Имя файловой переменной: имя типа файла.
```

Например:

```
Var
  F1: TFile Numb;
  F2: TSpisok Table;
```

F связывают с будущим файлом с помощью процедуры *assign*. Например: *assign (F, 'Dat.Dat')*. Непосредственное создание файла происходит при вызове процедуры *rewrite*. Например: *rewrite (F)*.

При программировании используются основные процедуры и функции языка для работы с данными, файлами. Например: прямоугольник описывается четырьмя целыми числами (Left, Top, Right, Bottom), задающими его удаление от осей координат или координатами своих двух угловых точек (TopLeft — верхней левой и BottomRight — нижней правой).

При описании точки в языке Object Pascal используется стандартный тип, предполагающий явное указание координат:

```
type
  TPoint = record
  X: integer;
  Y: integer;
end.
```

В системе есть более удобный набор инструментов для проектирования БД, чем процедуры и функции для работы с файлами. Это инструменты Data Base Desktop и DataBase Form Wizard (мастер конструирования формы).

В язык Object Pascal встроен стандартный генератор (счетчик) случайных чисел — функция *random* (случайный). Если не задавать аргумент, то в результате работы *random* выдаст случайное дробное число x в диапазоне $0 \leq x < 1$, если указать в скобках натуральное число *random(N)*, то будет выдано целое число из диапазона $0 \leq x < N$. И если использовать свойство счетчика с возможностью функции перевода числовых значений в строковые (*IntToStr*) и обратно (*StrToInt*), то очевидна большая эффективность ее при программировании различных вариантов в задачах, например, задав в поле редактирования *Edit1.Text := IntToStr(random(100))*.

При наличии многократных повторений однообразных операций массивы данных обычно подготавливаются заранее. Все эти массивы компьютер читает сам и организует процесс обработки с помощью циклических языковых конструкций:

- a) for... to...do или for... down to...do. // многократное повторение
- b) while i... do // цикл с предусловием;
- c) repeat ... until i... // цикл с постусловием.

Если при исполнении программы система обнаружит ошибку, то внизу окна модуля в специальном окошке будет выдано сообщение об ее обнаружении в определенной строке. Ошибки следует исправить и снова запустить программу на исполнение.

Для получения справочной информации об объекте, его нужно сначала выделить, а затем нажать клавишу F1. В системе есть также подсказчик кодов Code Completion — вспомогательное окно, предлагающее варианты кода, которыми можно завершить начатое предложение.

Рассмотрим примеры создания программ в среде Delphi.

Задача.

1. Постановка задачи.

Создать в форме проекта кнопку *Button*, шириной в определенных пределах и присвоить ей функцию появления надписи «Здравствуйте». Сохранить все три файла в отдельной папке.

2. Метод решения.

Программирование следует начинать со сценария — определения места расположения кнопки в форме, присвоения ей определенного значения, запуска программы на исполнение, сохранения всех трех файлов в отдельной папке.

3. Сценарий.

- a) кнопка должна располагаться ниже и справа от центра формы. Ей должно быть присвоено значение «Здравствуйте»;
- b) записать алгоритм суммы действий;
- v) запрограммировать разработанную форму проекта.

4. Алгоритм.

- a) На вкладке *Standard* найти компонент *Button* (кнопка). Щелкнуть на нем мышкой.
- b) Щелкнуть в форме справа, и ниже от центра возникнет кнопка с надписью *Button1*.

в) Два раза щелкнуть мышкой на кнопке *Button1* в форме. Откроется окно модуля *Unit1*, в котором должна появиться следующая заготовка:

```
Procedure TForm1.ButtonClick (Sender: TObject)
begin
end;
```

г) В заготовке написать команды звукового сигнала, диапазона размеров кнопки и присвоения кнопке значения «Здравствуйте».

д) Запустить программу на исполнение нажатием клавиши F9. В окне появится форма, содержащая исполняемый проект.

е) Сохранить все три файла в отдельной папке *Par* на диске D с помощью горячих кнопок или меню. Для этого нужно выбрать пункт меню *Save All* (сохранить все) или щелкнуть на кнопке пиктографического меню, и в открывшемся диалоговом окне сохранения файлов проекта открыть папку *Par*, созданную ранее, и в ней создать новую папку, например *Project1*. Затем, открыв ее, два раза нужно щелкнуть на кнопке «Сохранить»: первый раз для сохранения в ней файла с модулем *Unit1*, второй — для сохранения файла проекта *Privet.dpr*. В папке будут сохранены файлы *Privet.dpr*, *Unit1.pas* и *Unit1.dmf* — файл описания формы проекта (и еще несколько вспомогательных). Перед сохранением название файлов можно изменить, а расширение менять нельзя.

5. Программа.

Program Privet

```
Procedure TForm1.ButtonClick (Sender: TObject)
```

```
begin
```

```
beep; // звуковой сигнал
```

```
if Button1.Width < 30 then
```

```
Button1.Width := Button1.Width + 20
```

```
else Button1.Width := Button1.Width div 2; // целочисленное деление
```

```
Button1.Caption := 'Здравствуйте'; // присвоение значения кнопке
```

```
end;
```

```
end.
```

6. Форма и результат исполнения программы (рис. 17).

Задача 2.

1. Постановка задачи.

Создать проект «Буду учиться» при условиях: «И» и «ИЛИ» — «Поступил в колледж», «Я работаю».

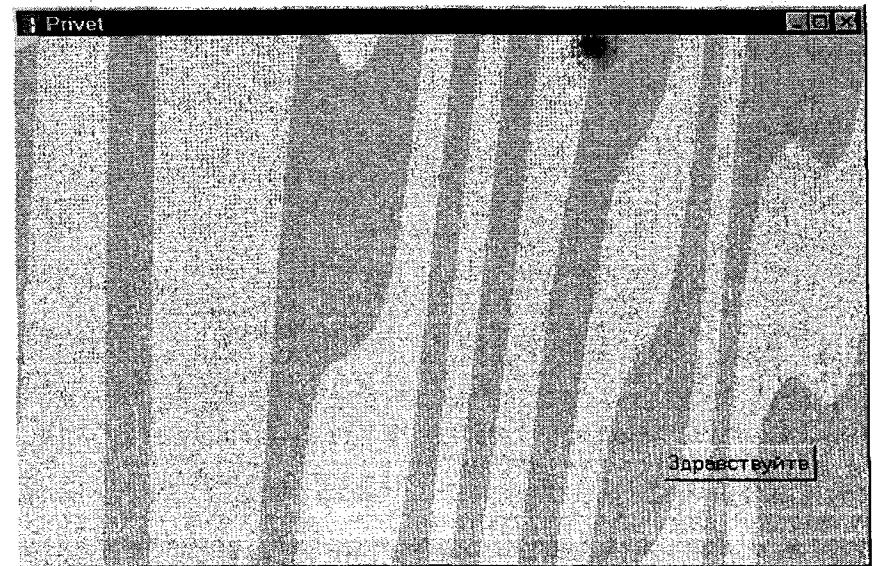


Рис. 17

2. Метод решения.

Программирование следует начинать со сценария — определения места расположения кнопок в форме, присвоения им значений, запуска программы на исполнение, сохранения всех трех файлов в отдельной папке.

3. Сценарий.

- кнопки должны располагаться слева и справа от центра формы.левой присвоить значение И, правой — ИЛИ;
- записать алгоритм суммы действий;
- запрограммировать разработанную форму проекта.

4. Алгоритм.

- На вкладке *Standard* найти компонент *CheckBox* (контейнер для отметок).
- Разместить два его экземпляра в форме: один — в верхней части формы, другой — в нижней.
- Дать им названия в свойстве *Caption* компоненты *CheckBox1* «Поступил в колледж», компоненты *CheckBox2* — «Я работаю».
- Разместить между контейнерами для отметок две кнопки *Button* — с названиями «И», «ИЛИ». Установить полужирный

шрифт (размер 20 для надписей), выбрав свойство Font в паспортах кнопок для установки параметров шрифта, щелкнуть в правом поле; появится кнопка с тремя точками. Два раза щелкнуть мышкой на этой кнопке, откроется диалоговое окно. Установить параметры.

д) Разместить внизу формы компонент *Label1* из вкладки Standard и дать название «Буду учиться» в свойстве Caption этого компонента. Для оформления надписи использовать свойства Font, Color. В паспорте *Label1* найти поле Visible (видимая) и установить значение false, используя раскрывающийся список значений.

е) Два раза щелкнуть на кнопку с надписью «И». В открывшемся окне Unit1 в заготовке процедуры TForm1.Button1.Click набрать команды условий: если оба контейнера отмечены, то сделать метку видимой, в противном случае — метку скрыть.

ж) То же выполнить для кнопки «ИЛИ»: откроется заготовка процедуры TForm1.Button2.Click. Скопировать (ctrl+c) ранее набранное выделенное условное предложение в буфер обмена и восстановить (ctrl+v) его из буфера в эту процедуру. В заготовке процедуры заменить and на or.

Теперь два раза щелкните мышкой на кнопке с надписью «ИЛИ».

з) Запустить проект с помощью клавиши F9. Сохранить файлы в своей папке под именем Project2.

5. Программа.

Program Project2

```

Procedure TForm1.Button1Click (Sender: TObject)
begin
    if CheckBox1.Checked and CheckBox2.Checked
    then Label1.Visible := true
    else Label1.Visible := false;
end;
Procedure TForm1.Button2Click (Sender: TObject)
begin
    if CheckBox1.Checked or CheckBox2.Checked
    then Label1.Visible := true
    else Label1.Visible := false;
end;
end.

```

6. Результат исполнения программы и модуль программирования задачи представлены на рис. 18.

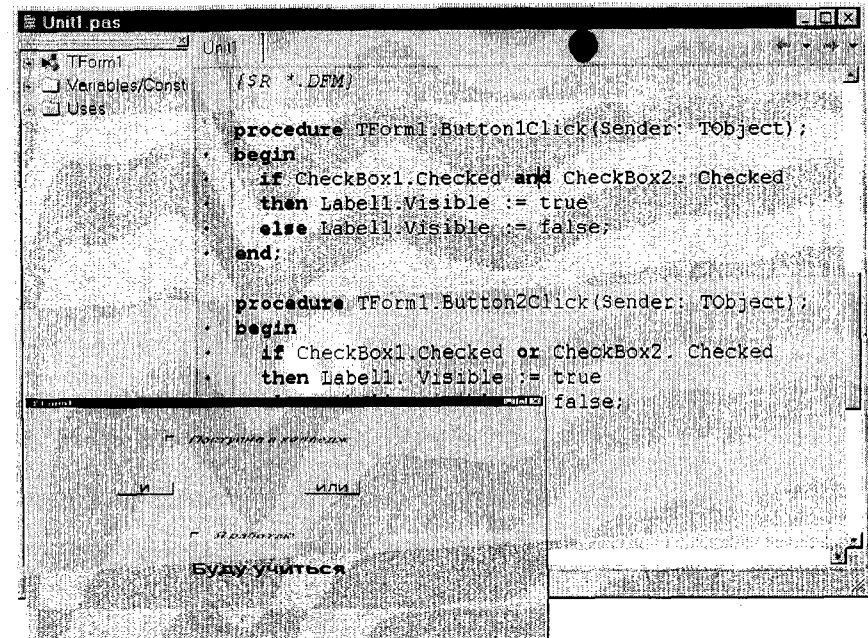


Рис. 18

Задача.

1. Постановка задачи.

Создать лица двух студентов с помощью инструментальных средств Delphi. Организовать и запрограммировать двухступенчатый диалог между ними, научив задавать вопросы и отвечать на них, открывая и закрывая рот (простейшая анимация).

2. Метод решения.

Программирование следует начинать со сценария — определения места расположения фигур, кнопок и меток в форме, присвоения им значений, запуска программы на исполнение, сохранения проекта в отдельной папке. Составить алгоритм и программу.

3. Сценарий.

- фигуры, кнопки и метки должны располагаться в выбираемых местах формы;
- записать алгоритм суммы действий;
- запрограммировать разработанную форму проекта.

4. Алгоритм.

а) Заготовить в форме проекта по 10 геометрических фигур для двух лиц с помощью компоненты Shape (фигура) вкладки Addition. Прямоугольники изменить на фигуры, которые позволяют осуществлять компоновку лица:

- для Shape1 — форма 1-го лица (Студент), выбрать St Round Square — квадрат с округленными углами; для Shape11 — форма 2-го лица (Студентка), выбрать St Ellipse — эллипс;
- для Shape2, Shape3 и Shape12 и Shape13 — форма бровей 1-го лица и 2-го лица, выбрать St Rectangle — прямоугольник;
- для Shape4 и Shape5 и для Shape14 и Shape15 — форма глаз, выбрать St Circle — окружность;
- для Shape6, Shape7 и Shape16 и Shape17 — форма зрачков, выбрать St Circle — окружность;
- для Shape8 и Shape18 — форма носа, выбрать St Ellipse — эллипс;
- для Shape9 и Shape10 и для Shape19 и Shape20 — форма губ, выбрать St Ellipse.

//st — тип фигуры (от слов shape и type).

С помощью мыши выбрать целесообразные размеры элементов лица. С помощью палитры установить цвета. Цвет контура можно задать, нажав на + в составном свойстве Pen (Контур) свойства Color, и выбрать цвет в развернувшемся списке. Аналогично можно задать закраску контура свойством Brush (Кисть).

б) Процесс открывания рта для каждого лица выполнить в два этапа: закрытие рта и открытие рта через интервал в 400 мс. Чтобы выполнить первый этап, следует разместить в форме кнопки Button1, задав ей заголовок: «Студент», и Button2, задав ей заголовок: «Студентка». Открыть обработчик событий ON CLICK. Запрограммировать процедуры открытия и закрытия рта с помощью таймера.

Реакцию на срабатывание таймера по прошествии 400 мс написать в процедуре TForm1.Timer1.Timer, заготовка которой откроется, если два раза щелкнуть мышкой на компоненте формы Timer1.

в) Запустить программу на выполнение, файлы сохранить в папке Project 3.

г) Создать специальное окно для ввода вопроса с помощью вкладки Standard и компоненты Edit1, разместив ее в форме около рта Студентки. Слово Edit1 в этом окошке убрать, удалив содержимое в поле Text его паспорта.

д) Для ответов использовать компоненту Label1, разместив ее над нижней губой Студента. Очистить поле Caption в паспорте Label1 и выбрать нужные характеристики шрифта и цвета.

е) Добавить коды двухступенчатого диалога в форме обработчика событий.

Диалог:

Студентка: «Привет». Студент: «Здравствуйте».

Студентка: «Что такое Delphi?». Студент: «Неужели не знаешь?»

ж) В сгенерированной среде Delphi объявить flag булевской глобальной переменной. Инициализировать глобальные переменные: s — как строковую, f — как булевскую.

Запустить программу на выполнение (F9). Сохранить проект. Перед сохранением изменить название «Project3» на «Диалог».

При необходимости следует устранить дефекты диалога с помощью операторов присвоения метке — пустой строки и отключения таймера.

5. Программа.

Program Project3

var

s: = edit.text; // s:=string присваивать значение из окна Edit1

f: boolean // состояние флага

Procedure TForm1.Button1Click (Sender: TObject)

begin

Shape9.Height: = 238; // закрыт рот

Shape19.Height: = 161; // открыт рот

Timer1.Enabled: = true; // запуск таймера

end

Procedure TForm1.Button2Click (Sender: TObject)

begin

if s= 'Привет' then

f: = true;

Label1.Caption: = 'Здравствуйте';

end;

if s='Что такое Delphi?' and f then

begin

Label1.Caption: = 'Неужели не знаешь?';

f: = false;

end;

Procedure TForm1.Button3Click (Sender: TObject)

Begin

Shape9.Height: = 172; // открыт рот

Shape19.Height: = 233; // закрыт рот

Timer1.Enabled: = false; // и отключить таймер

end;

end.

6. Результат исполнения программы (рис. 19, 20).

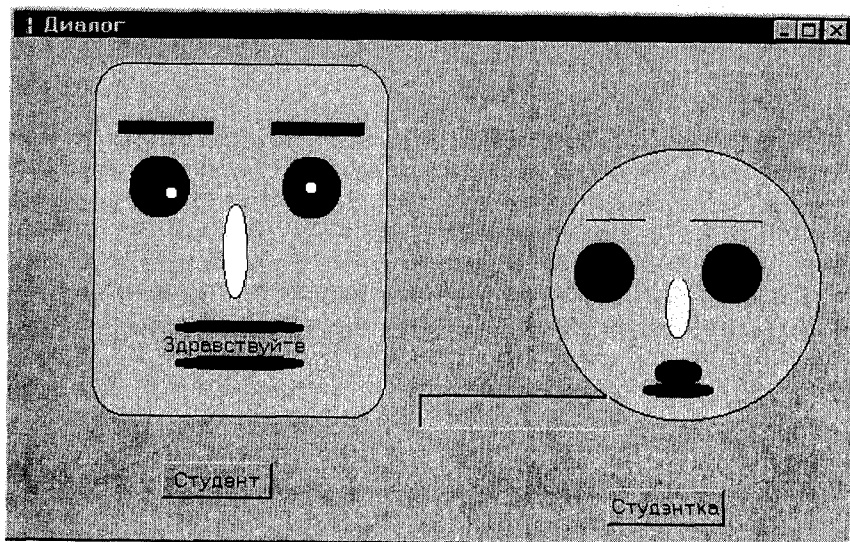


Рис. 19

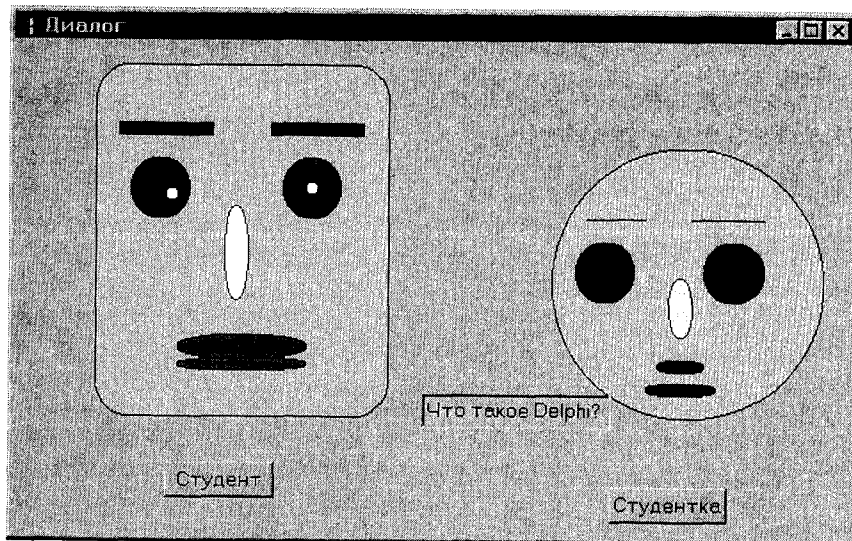


Рис. 20

Можно написать программу и так:

```

procedure TForm1.Button2Click(Sender: TObject);
begin
    Timer1.Enabled := True;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
    Timer2.Enabled := True;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
    Shape9.Top := 172;
    Label1.Visible := False;
    if Temp = 1 then
        Edit1.Text := 'Привет!'
    else Edit1.Text := 'Что такое Delphi?';
    Inc(Temp);
    Shape10.Top := 238;
    Timer1.Enabled := False;
end;

procedure TForm1.Timer2Timer(Sender: TObject);
begin
    Edit1.Clear;
    Shape19.Top := 161;
    Label1.Visible := True;
    if Temp = 1 then
        Label1.Caption := 'Здравствуйте'
    else
        Label1.Caption := 'Неужели не знаешь?';
    Shape20.Top := 233;
    Timer2.Enabled := False;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
    Temp := 1;
end;
end.
```

Контрольные вопросы

1. Что такое инструментальные средства Delphi?
2. Какие функции выполняет инспектор объектов (Object Inspector)?
3. В каких формах создается и исполняется проект?
4. Какие виды и типы диалога вы знаете?
5. С помощью какой процедуры создается список случайных фраз для диалога?

Глава 4**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
ДЛЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
И АВТОМАТИЗИРОВАННЫХ СИСТЕМ**

Производство ПО для вычислительной техники (ВТ) и автоматизированных систем (АС) — крупнейшая отрасль мировой экономики, в которой заняты миллионы людей.

Человек всегда стремился заменить трудоемкие процессы физического и умственного труда работой технических средств. Автоматизация зарождалась, когда для функционирования измерительных, регулирующих и исполнительных устройств подключали внешний источник энергии. Постепенное развитие автоматизируемых устройств привело к организации постов контроля и дистанционного управления, широкому применению автоматических регуляторов. Объединение таких постов привело к созданию центральных щитов управления, а затем центральных пунктов управления. Введение унифицированных измерительных и управляющих сигналов, передаваемых на расстояние, позволило переработку информации территориально отделить от технологического процесса. Она концентрировалась в специализированных пунктах с соответствующим оборудованием для ее обработки и осуществления управляющих воздействий на объекты. Но управлять большими объектами трудно, поэтому возникла проблема автоматизации собственно управления, т. е. процесса принятия решений. Это потребовало применения современных экономико-математических методов и новых технических средств. Появились автоматизированные системы управления (АСУ), человеко-машинные системы, реализующие автоматизированный сбор и переработку информации, необходимой для принятия решений по управлению объектом. В этих системах широко используются различные средства автоматизации, включая вычислительную технику. ЭВМ выполняет практически все процессы обработки первичной информации и централизованного контроля. А режим диалога человека с машиной обеспечивает его участие в процессе решения задач и принятия ответственных решений.

Первоначально АСУ строились на основе больших ЭВМ и предполагали централизованную обработку информации, поступающую с различных абонентских пунктов. Затем их создавали, используя мини- и микро-ЭВМ. Но особенно бурное развитие автоматизация во всех сферах человеческой деятельности получила с появлением и распространением ПК и ЛВС.

Сам по себе ПК не обладает знаниями ни в одной области своего применения, все эти знания сосредоточены в выполняемых на нем программах. Принцип прямого цифрового управления применяется во многих современных промышленных компьютерных системах: от управления температурой в пассажирском салоне до высокотехнологичных производств. Сегодня простейший регулятор экономически выгодно создавать на основе цифровых устройств. Кажется, что системы управления химическим производством или движением в большом городе имеют мало общего с роботом по переноске деталей. Однако во всех АС можно выделить одинаковые блоки: сбор данных, управляемые функции, обратная связь, обмен данными, взаимодействие с человеком-оператором.

Автоматизация производства и автоматизация управления идут одновременно и во взаимосвязи друг с другом. Большое внимание уделяется АСУ различными объектами. *Автоматизированные системы управления* создают для оптимального управления в различных сферах человеческой деятельности. Оптимальное управление — это выбор такого варианта управления, при котором достигается экстремальное (максимальное или минимальное) значение критерия, характеризующего качество управления (критерия эффективности). Это система, в которой для получения, обработки информации и управления используются различные устройства и ЭВМ. Однако такие важные функции, как определение и корректировка целей и критериев эффективности системы, уточнение методов решения задач, принятие окончательных решений и другие, выполняются человеком (группой людей). Нельзя АСУ путать с САУ — системой автоматического управления, в которой управление полностью выполняют технические средства, без непосредственного участия человека в этом процессе.

Если раньше программирование сводилось к написанию кода, то теперь оно включает сбор и анализ требований, проектирование, кодирование, дизайн, документирование, реализацию, управление изменениями, тестирование, а также менеджмент (управленческий механизм). И сам процесс получил название, более соответствующее его сути — разработка ПО.

Программное обеспечение для ВТ и АС — это совокупность программ для обеспечения работы комплекса технических средств и реализации целей и задач АС различного направления. ПО тесно связано с математическим обеспечением (МО) и составляется на его базе. Все программное обеспечение поставляется на флоппи-дисках, лазерных дисках (CD, DVD) или через международную сеть Интернет. Иногда программный продукт может стоить дороже самого компьютера.

Весь комплекс ПО делится на системные и пользовательские программы. Системное ПО выполняет функции организации всех частей компьютера и подключенных к нему внешних устройств и управления ими. Пользовательские программы служат для выполнения конкретных задач во всех сферах человеческой деятельности.

В программной инженерии понятие жизненного цикла (ЖЦ) ПО является одним из базовых. Он определяется как период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации. ЖЦ ПО регламентирует нормативный документ, которым является международный стандарт ISO/IEC 12207:1995 «Information Technology — Software Life Cycle Process» (ISO — International Organization for Standardization — Международная организация по стандартизации; IEC — International Electrotechnical Commission — Международная комиссия по электротехнике). В данном стандарте ПО (или программный продукт) определяется как «набор компьютерных программ, процедур и, возможно, связанной с ними документации и данных».

Первоначально (в 1970-е гг.), в России создание ПО регламентировалось стандартами ГОСТ ЕСПД (Единая система программной документации — серия ГОСТ 19.XXX). Эти стандарты устарели. В настоящее время процессы создания АС, в состав которых входит и разработка ПО, регламентированы стандартами ГОСТ 34.601—90, ГОСТ 34.602—89, ГОСТ 34.603—92, входящих в сборник стандартов «Информационная технология. Комплекс стандартов на автоматизированные системы». Однако и в этих стандартах некоторые положения устарели, а другие не отражены, так как развитие программной инженерии стремительно развивается. В результате для многих проектов создают комплексы нормативных и методических документов, регламентирующих процессы создания конкретного прикладного ПО. Поэтому целесообразно ориентироваться на международные стандарты.

Стандартизация и типизация, использование типовых проектных решений позволяют не только выпускать программные продук-

ты высокого качества, но также сократить трудоемкость разработок и время внедрения.

Создано много стандартных программ. *Стандартные программы* — общеупотребительные программы, разработанные в соответствии с ГОСТ и построенные так, что их можно включать в состав пакета прикладных программ для решения разных задач. Библиотеки стандартных программ формируются и содержатся на магнитных носителях под определенными именами.

К настоящему времени создано много пакетов прикладных программ (ППП), использующихся как для обеспечения работоспособности ВТ, так и в работе АС. В общем смысле ППП — структурно-сложные системы программ, предназначенные для решения задач определенного класса. Они, как правило, состоят из программ общего и функционального назначения. Первые реализуют типовые режимы работы ВТ, вторые обеспечивают обработку данных. ППП для вычислительного процесса и ведения информационных баз создают на базе проблемно-ориентированных АС создания ПО. Иногда ППП — набор подпрограмм, объединенных управляющей программой и предназначенный для решения задач в какой-либо области знаний.

Контрольные вопросы

1. Что такое АС?
2. Что такое АСУ?
3. Что включает понятие ПО для ВТ?
4. Что включает понятие ПО для АС?
5. Что такое ППП?

4.1. Программное обеспечение для вычислительной техники. Системные программные средства

Программы для работы аппаратной части компьютера создавались в течение многих десятилетий. ПО для ВТ можно разделить на внутреннее, обеспечивающее нормальную работу ЭВМ, и внешнее, позволяющее потребителю использовать его как стандартное при решении на ЭВМ необходимых задач наиболее простым образом. ПО включает в себя систему программирования, ОС, ППП.

Внутреннее ПО состоит из базовых — эксплуатационных программ (*тестовых и диагностических*), проверяющих исправность

оборудования ЭВМ, систем программирования и ОС. Внутреннее ПО тесно связано со структурой ЭВМ и реализует возможности, заложенные в аппаратуре.

Система программирования предназначена для автоматизации процесса программирования и содержит трансляторы алгоритмических языков высокого уровня, редактор текстов программ, библиотеки полезных программ, отладчики и другие вспомогательные программы. Система программирования содержит целый набор языков, которые в совокупности исчерпывают все типы задач, решаемых в разных системах. Кроме того, она включает библиотеки стандартных программ.

Важным классом системных программ являются *драйверы*, расширяющие возможности ОС, позволяя работать с тем или иным внешним устройством, обучая ее новому протоколу обмена данными и т. д. С помощью драйверов можно подключить к компьютеру новые устройства или нестандартно использовать имеющееся оборудование. Например, на мыши имеется центральная кнопка, которая часто не используется, но если установить специальный драйвер, то она будет функционировать по определенному назначению. Драйверы для различных ОС часто поставляются вместе с новыми устройствами и контроллерами (электронные схемы для управления внешними устройствами).

К системным программам относится большое количество *утилит* — программ вспомогательного назначения, например программы резервирования (копирования), антивирусные, архиваторы, русификаторы, оптимизации дисков, динамического сжатия дисков, ограничения доступа и др. Часто утилиты объединяют в такие комплексы, как, например, Check-It, Norton Utilities, SiSoft Sandra, Nuts&Bolts, которые включают в себя утилиты по проверке жестких и флоппи-дисков, микропроцессора, оперативной памяти, модемов, принтеров, ускорения запуска приложений, восстановления ранее удаленных файлов и т. д.

Антивирусные программы отслеживают распространение всех видов вирусов (специальных вредных саморазмножающихся программ) на компьютере и по возможности лечат зараженный объект: удаляют его или запрещают доступ к нему. Известны антивирусы Norton Antivirus фирмы «Symantec». Но самыми лучшими считаются антивирусные программы, выпущенные российскими производителями: АО «Диалог Наука», «Лаборатория Касперского». Они выпускают такие известные во всем мире антивирусные программы, как Dr.Web 32, Adinf 32, AVP.

Программы-архиваторы позволяют «упаковывать» информацию, сжимать ее на дисках, т. е. создавать копии файлов гораздо меньшего объема. Можно объединять копии нескольких файлов в один архивный. Например, универсальный архиватор WinRAR позволяет создавать архивные файлы с расширением arj, rar, zip

Для IBM-совместимых компьютеров, работающих с популярными языками (Си, Си++, Паскаль, Бейсик), созданы системы программирования, работающие в среде ОС DOS и Windows. В последнее время появились системы программирования на языке Java, который позволяет создавать программы, вызываемые при просмотре Web-страниц в глобальной электронной сети Интернет. Создаются и системы программирования для создания приложений типа клиент-сервер, позволяющие быстро создавать информационные системы как для подразделений крупных предприятий, так и для самих предприятий. В них содержатся средства для создания пользовательского интерфейса, описание процедур обработки данных, подготовки для выполнения типовых действий по обработке данных и т. д. Эти системы позволяют эффективно работать с различными СУБД.

Операционная система (ОС) — комплекс программ, управляющих всеми узлами ЭВМ и внешними устройствами и обеспечивающих требуемый режим обработки данных. ОС может быть разделена на управляющую и обрабатывающую части. Управляющая часть обеспечивает решение задачи в требуемом режиме, обрабатывающая — транслирует содержание задачи, записанной на некотором языке программирования, редактирует программные модули и генерирует необходимую конфигурацию вычислительной системы в целом. ОС реализует функции управления данными, заданиями и задачами, обеспечивает защиту данных

Основной принцип построения гибких и сложных ОС заключается в выделении типовых процедур и оформлении их в виде отдельных стандартных блоков. Такой принцип называется модульным. *Программный модуль* — программный блок, реализующий определенные функциональные возможности и рассчитанный на стандартные формы связи. Наиболее крупными программными блоками ОС являются *супервизор и монитор*.

Супервизор постоянно находится в оперативной памяти, координирует поток задач через систему, распределяет ее ресурсы, планирует все операции ввода-вывода, сообщает оператору о неисправностях, осуществляет обслуживание по таймеру. Монитор обеспечивает управление решением задач на ЭВМ в различных режимах, воспринимает от рабочих программ или от оператора управляющие

команды (директивы) и организует их выполнение. Постоянно в оперативной памяти находится только часть монитора — резидент, остальная часть хранится во внешней памяти и вызывается по мере необходимости. В некоторых ЭВМ супервизор и монитор образуют программу-диспетчер.

На компьютерах типа IBM PC чаще всего применяются следующие ОС: MS DOS фирмы «Microsoft» или совместимые с ней PC DOS фирмы IBM и Novell DOS фирмы «Novell», Windows фирмы «Microsoft», OS/2. Самая первая операционная система — MS DOS (Disk Operation System) фирмы «Microsoft» была 16-разрядной, однозадачной и могла общаться с пользователем посредством безликой командной строки на экране. До сих пор выпускают такие модифицированные версии DOS, как PC-DOS 2000 фирмы IBM, хотя их редко устанавливают на компьютерах как основную ОС. Почти не используют такие первые версии графических ОС, как Windows 3.x (конец 1980-х гг.), Windows 95. В настоящее время на PC устанавливают 32-разрядные ОС фирмы «Microsoft»: Windows 98, Windows 2000, Windows Millennium Edition и т. д.

Популярный класс системных программ — *операционные оболочки*, обеспечивающие удобный и наглядный способ общения пользователя с компьютером. Операционные оболочки представляют собой надстройки над дисковыми ОС. Интерфейс оболочки позволяет пользователю удобно и эффективно работать с ОС, осуществлять диалог с ЭВМ. В оболочках иерархическая структура (организация) хранения папок (каталогов) файлов на диске обеспечивает быстрый доступ к ним. Для работы с файлами имеются специальные программы. Примером долговременного использования такой оболочки может служить Norton Commander для MS DOS. Известны также операционные оболочки, как Volkov Commander, PowerDesk, DOS Navigator, Disco Commander, Far, Windows Commander и др.

Сетевые операционные системы созданы для обслуживания локальных сетей. Они служат для диспетчерского управления другими компьютерами, находящимися в локальной сети. Сетевыми ОС распоряжаются администраторы сети. Такими ОС являются: MS Windows NT, Novell Net Ware, LAN WorkPlace и др.

Внешнее (общесистемное) ПО состоит из:

- библиотеки типовых программ: ввода, контроля, сортировки, корректировки, дублирования, обработки единиц информации, поиска и вывода информации;
- программ решения конкретных задач АС;
- системной диспетчерской программы.

Из разрабатываемых стандартных ППП можно выделить 3 группы:

- расширяющие возможности ОС (обеспечивают работу много-машинных комплексов, диалоговых систем, работу в реальном времени, удаленную пакетную обработку);
- общего назначения (для дисплеев, графопостроителей, систем программирования, обработки матриц, моделирования, решения задач теории массового обслуживания и т. п.);
- ориентированные на работу в АСУ (для общецелевых систем обработки банков данных, информационно-поисковых систем), систем обработки документов.

Для различных применений на IBM PC разработаны и наиболее широко используются многочисленные прикладные программы:

- подготовки текстов на компьютере — редакторы, формтеры, конверторы текстов и текстовые процессоры;
- подготовки документов типографского качества — настольные издательские системы;
- создания графических изображений — графические редакторы;
- обработки табличных данных — табличные процессоры;
- распознавания символов;
- обработки массивов информации — системы управления базами данных;
- подготовки презентаций;
- экономического назначения: бухгалтерские, финансового анализа, правовые информационно-поисковые системы и т. д.;
- статистической обработки данных, линейной алгебры, дифференциального и интегрального исчисления, нахождения корней уравнений и экстремума, методов численного анализа и т. д.;
- черчения и конструирования различных предметов и механизмов — системы автоматизированного проектирования;
- обучающие, компьютерные игры, электронные справочники и т. д., использующие средства мультимедиа;
- виртуальной реальности;
- телекоммуникационных технологий (WWW, электронная почта и др.)

Текстовые редакторы и процессоры позволяют подготавливать документы быстро и удобно, выполнять много функций редактирования и форматирования документов. Например, использовать различные шрифты, абзацы разной формы, делать автоматический перенос слов и нумерацию страниц, сноски, оглавления, проверять

правописание. Текстовые процессоры включают в себя и ряд таких сложных функций, как разбиение текста на столбцы, вставка рисунков, таблиц, диаграмм и т. д.

Редактор текстов обеспечивает ввод, изменение и сохранение символьного текста. Объектом редактирования является фрагмент текста (от одного символа до всего текста), т. е. область, указанная (выделенная) пользователем. К функциям редактирования относят добавление, удаление, перемещение, копирование, поиск и контекстную замену. Текстовый редактор предназначен в основном для подготовки текстов программ, т. к. они не требуют форматирования. Используются в экранном редакторе (Norton Editor, фирмы «Peter Norton Computing Inc.», Multi Edit фирмы «American Cybernetics Inc.» и др.). Редактор текстов входит, как правило, в Турбо-системы — инструментальные средства для создания, компиляции, отладки и выполнения программ на языках высокого уровня.

Если текст подготавливается на естественном языке для печати, то средств редактора текстов уже недостаточно, так как требуется работа с абзацами, страницами, разделами, разными шрифтами и т. д., т. е. форматирование (оформление). Тогда в зависимости от сложности документа используют *формтеры*, которые для представления текста используют только стандартные коды: конец строки, перевод каретки, конец страницы и др., текстовые процессоры и настольные издательские системы.

С помощью *текстовых процессоров* (например, MS WORD 98 (рис. 21)) можно выполнять набор текста, его редактирование и форматирование, сохранение и архивирование, печать, создавать документы, включающие различные виды информации: текст, таблицы, формулы, рисунки, иллюстрации.

Недостаток текстовых процессоров в том, что каждый из них имеет свою структуру данных для представления текста. Из-за этого текст, подготовленный в одном текстовом процессоре, нельзя прочитать, отредактировать и отформатировать в другом. Для совместности документов, созданных с помощью разных текстовых процессоров, применяют *конверторы* — программы, преобразующие формат выходного файла с одного процессора в формат файла другого текстового процессора.

Настольные издательские системы предназначены для верстки документов, т. е. размещения текста по страницам, разбиение его на колонки, вставки рисунков, использования различных изобразительных эффектов. Такие системы позволяют подготовить документы к изданию, эффективно обеспечивая полиграфическую точность, цвета и управление макетом.

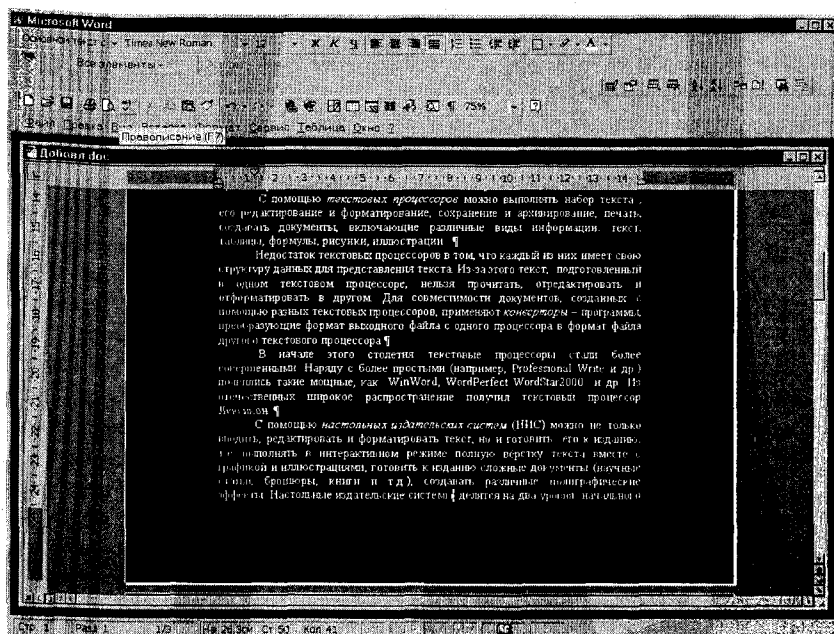


Рис. 21

С помощью *настоенных издательских систем* (НИС) можно не только вводить, редактировать и форматировать текст, но и готовить его к типографскому изданию, т. е. выполнять в интерактивном режиме полную верстку текста вместе с графикой и иллюстрациями, готовить к изданию сложные документы (научные статьи, брошюры, книги и т. д.), создавать различные полиграфические эффекты. Настольные издательские системы делятся на два уровня: начального и профессионального. НИС начального уровня (например, MS Publisher и др.) предназначены для эпизодического создания бюллетеней, открыток и т. п. НИС профессионального уровня предназначены для издания сложных документов, иллюстрированных журналов (например, PageMaker, Corel Ventura 8 и др. (рис. 22)).

Современные системы подготовки текстовых документов, обладающие дружественным пользовательским интерфейсом, дают возможность в разных областях профессиональной деятельности быстро создавать необходимые хорошо напечатанные и оформленные документы и размножать их в необходимом для пользователей количестве.

Графические редакторы предназначены для создания рисунков, схем, иллюстраций. Векторные редакторы используют для создания

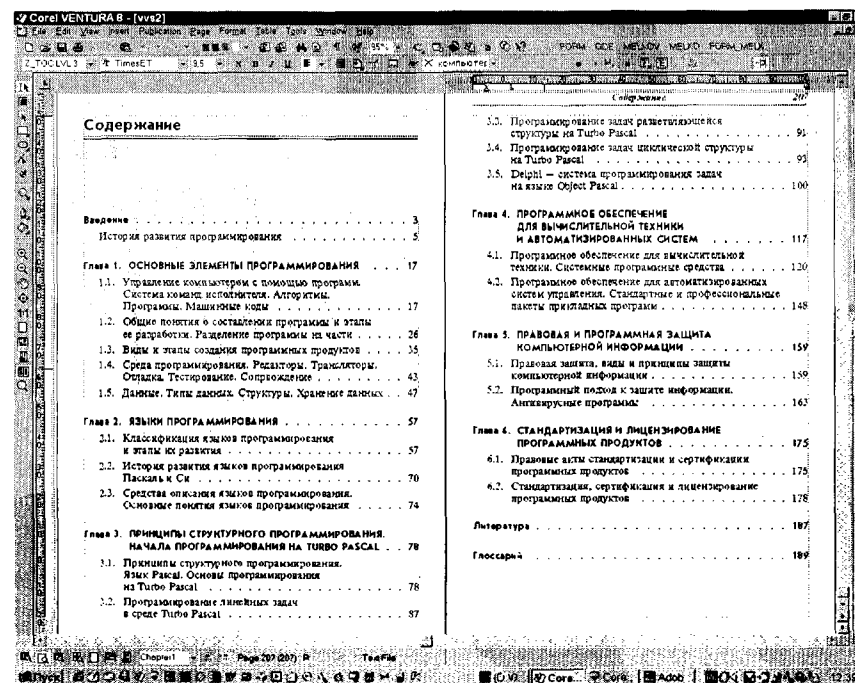


Рис. 22

более точных чертежей, схем и рисунков, а растровые — для обработки художественных изображений, фотоизображений, создания красочных иллюстраций.

Системы подготовки графических материалов — инструментальные программные средства, предназначенные для создания и модификации графических объектов. Как правило, эти системы применяют для научной и образовательной, коммерческой и иллюстративной графики.

Научная графика широко используется в виде отдельных модулей приложений ОС (например MS Equation — редактор формул и др.) и их встроенных функций (например векторный графический редактор Word и др.), в автоматизированных системах проектирования (например AutoCad и др.), в геоинформационных системах.

Коммерческая (деловая) графика используется для отображения различных экономических, финансовых и социальных показателей, табличных данных. В виде иллюстраций коммерческой графики могут быть диаграммы разных видов, в том числе гистограммы — диа-

граммы, показывающие изменение данных за определенный период времени, графики — отображают изменение данных за равные периоды времени, линейчатые диаграммы — отражают соотношение отдельных компонентов, круговые — показывают абсолютные величины данных и процент от общей суммы и др.

Но более всего распространены системы иллюстративной графики, использующиеся для различных целей практически в любой сфере деятельности. Графические редакторы и процессоры можно разделить на два вида: системы векторной и растровой графики, каждая из которых, в свою очередь, — на системы, предназначенные для создания графических изображений, и системы, в которых выполняют в основном манипулирование (в том числе редактирование) графикой (рис. 23).

Системы подготовки графических материалов могут быть реализованы как самостоятельные программы построения графических изображений (например Corel Draw, MS Photo и др.), пакеты программ для создания презентаций (MS Power Point, MacroMedia Flash и др), функции офисных приложений.

Табличные процессоры позволяют выводить на экран прямоугольные таблицы, в ячейки (клетки) которых можно вводить числа, тексты, формулы для расчетов по имеющимся данным. Значения элементов таблиц могут перевычисляться по задаваемым формулам. По данным таблиц можно строить графики. Мощные процессоры позволяют на основе расчетов создавать сложные документы с диаграммами, рисунками, создавать БД.

Документ в Excel представлен в виде таблицы (столбцы и строки — поля и записи). Пересечения столбцов и строк образуют сетку ячеек, в которые помещают данные любого вида, но по преимуще-

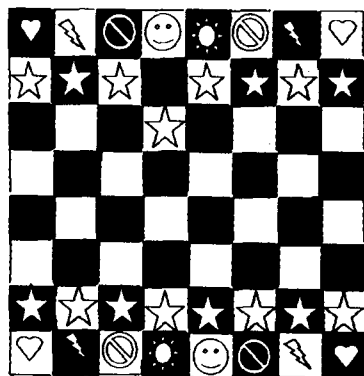


Рис. 23

ству числовые, так как наиболее эффективно в электронных таблицах обрабатывать массивы числовой информации. Таблица должна быть логично построена, удобна для чтения и экономична, т. е. — с однородными характеристиками, содержать лаконичные тексты и небольшие иллюстрации, не допускать лишних граф. Структура рабочего окна Excel представлена на рис. 24. Она состоит из строки заголовка, строки меню, пиктографического меню (инструментальные панели), строки ввода (редактирования), рабочего листа, линейек прокрутки, делителей окна, строки сообщений.

Современные табличные процессоры имеют большие возможности для эффективной работы пользователя: контекстные подсказки, вызываемые из контекстного или пиктографического меню, справочную систему, многовариантность выполнения операций, рабочие папки, средства для оформления и модификации экрана и таблиц, средства оформления и вывода таблиц на печать, средства оформления рабочих листов, шаблоны, связывание данных с помощью относительной и абсолютной адресации (работа с несколькими таблицами, связь нескольких листов, консолидация рабочих листов

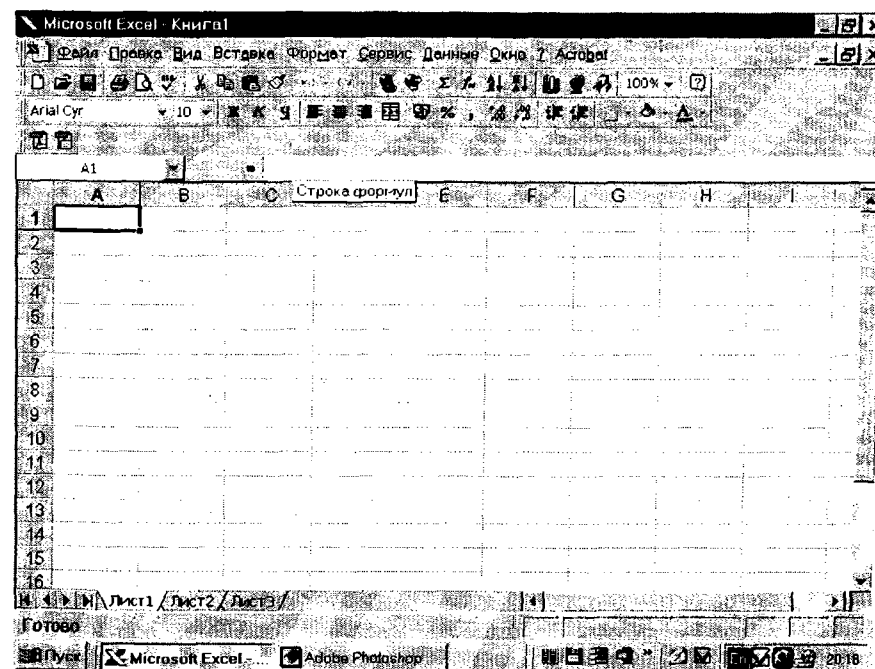


Рис. 24

и книг), вычисления, деловая графика (диаграммы и способы их формления, вставка планок погрешностей, построение тренда и др.), выполнение функций реляционных (табличных) БД, моделирование (подбор параметров и использование методов оптимизации для нахождения оптимальных решений разных задач), макропрограммирование (использование встроенного языка программирования макрокоманд для часто повторяемых действий) (рис. 25, 26). В электронных таблицах можно обрабатывать данные, в следующих форматах: общий (для текстовой и числовой информации), числовой, текстовый, денежный, календарный и временной, процентный, финансовый и др. Ячейки таблицы могут содержать как числовые и текстовые константы, так и выражения (арифметические или логические) в виде формул. В Excel, например, имеется также множество (около 400) встроенных функций для различных расчетов, работы с БД, списками и др. Документы, созданные в электронных таблицах, используют и как самостоятельные, и как составные части других сложных документов (рис. 27).

Программы распознавания символов позволяют вводить с помощью сканера напечатанные тексты, делая ненужным утомительный и трудоемкий ввод текста с клавиатуры. Качество распознавания текста зависит от качества вводимого.

Системы управления базами данных (СУБД) — дают возможность управлять большими информационными массивами информации — БД. БД — это поименованная структурированная совокупность данных, относящихся к конкретной предметной области. СУБД предназначены для интерактивного применения и включают средства программирования. Есть СУБД для разработки небольших информационных систем, а есть — для сложных типа клиент — сервер. В этом случае база расположена на большом компьютере-сервере, который принимает от программ, выполняемых на других компьютерах-клиентах, запросы на получение информации из БД.

СУБД — совокупность методов, языковых и программных, средств, предназначенных для создания, ведения и использования БД. Методы и средства СУБД позволяют осуществить сбор, регистрацию, хранение, упорядочение, поиск, выборку по запросам и представление информации БД. С помощью СУБД можно создавать и хранить большие массивы данных и манипулировать ими. БД создают в соответствии с моделями описания данных и их свойств в БД. Выделяют реляционную, иерархическую и сетевую модели построения БД, а соответственно, и СУБД.

Иерархические БД строят в соответствии с моделью описания объектов, входящих в предметную область как совокупности иерар-

Переменные				
Имя	Краска 1	Краска 2	Доход	
	2	1		
Ограничения				
Ресурс			Расход	Запасы
A	1	2		3
B	3	1		3

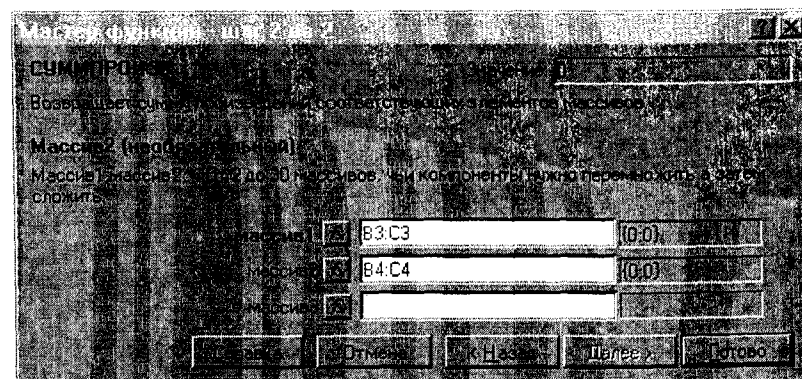
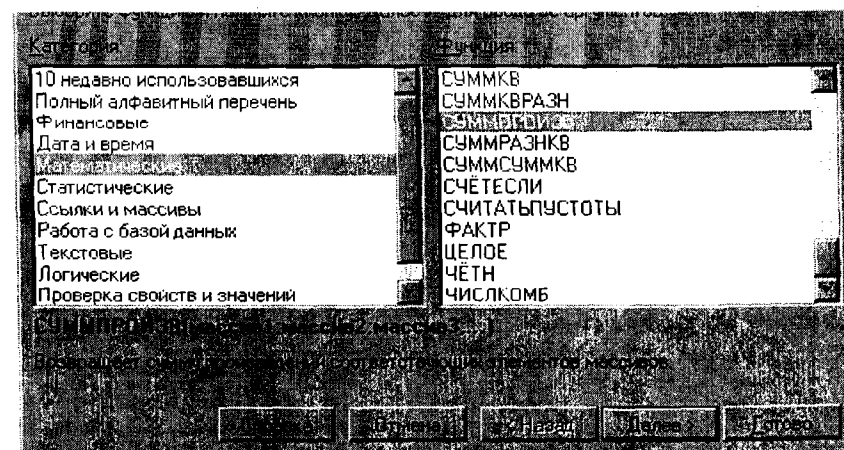


Рис. 25

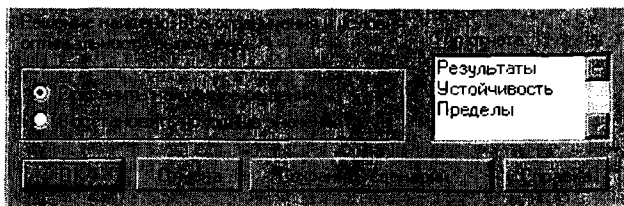
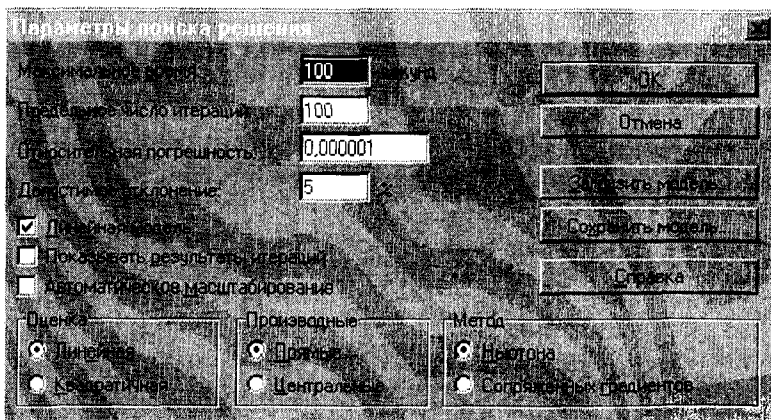
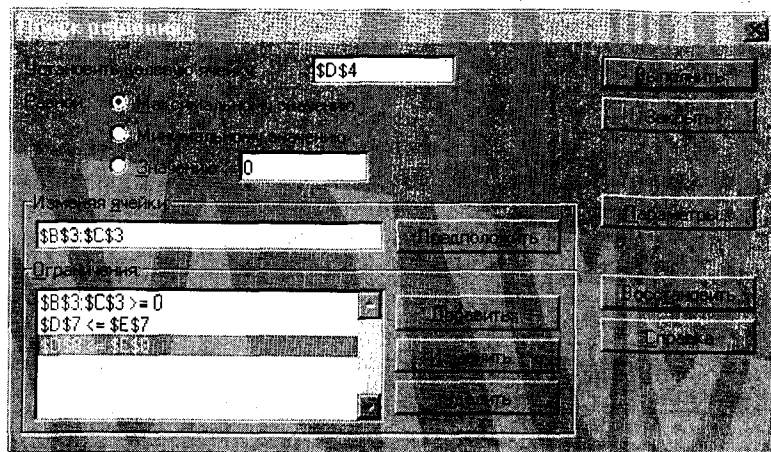


Рис. 26



Рис. 27

хически (по уровням подчинения) взаимосвязанных объектов и их атрибутов (свойств), где отношения построены по принципу «один ко многим».

Реляционные БД строят в соответствии с моделью описания объектов, входящих в предметную область как совокупности взаимосвязанных атрибутов, т. е. свойств, находящихся в определенных отношениях. Отношения построены по принципу «один ко многим». Реляционная модель БД представляется чаще всего в виде взаимосвязанных таблиц, состоящих из полей (столбцов) и записей (строк). В каждой таблице выбирается ключ — поле, однозначно определяющее уникальное значение каждой записи, по которому осуществляют поиск и выборку данных.

Сетевые БД строят в соответствии с моделью описания объектов, входящих в предметную область как совокупности взаимосвязанных объектов и их атрибутов (свойств), где отношения построены по принципу «многие ко многим» и «многие к одному».

Создаются СУБД общего и специализированного назначения. СУБД общего назначения строят, как правило, для определенной ОС (например MS Access является приложением ОС Windows) и модели ЭВМ. При невозможности использовать СУБД общего назна-

чения разрабатывают специализированные, обеспечивающие дополнительные функции, которых нет у стандартных СУБД, или они не устраивают пользователей по таким характеристикам как скорость поиска информации по запросам, время создания отчетов, число параллельных обращений к данным и др.

Следует также отметить, что разнообразные офисные пакеты позволяют для создания больших и сложных документов, содержащих информацию различного вида использовать технологию OLE. Технология OLE (Object Linking and Embedding) — технология связывания и внедрения объектов, позволяющая переносить (внедрять) из одних в другие файлы информацию различного вида (текст, графику, таблицы, звуковой клип и т. д.), создавая связанные составные документы (файлы), содержащие совокупность этой информации. В качестве программы-клиента выступает программа, собирающая составной документ, а программы-сервера — программа, в которой ранее был создан внедряемый (переносимый) объект.

Системы подготовки презентаций могут создавать слайды для презентаций, располагая на них красивые рисунки, надписи, диаграммы, включать звук и анимацию, показывать их с помощью компьютера в любом порядке в ручном или автоматическом режимах (рис. 28—34).



Рис. 28

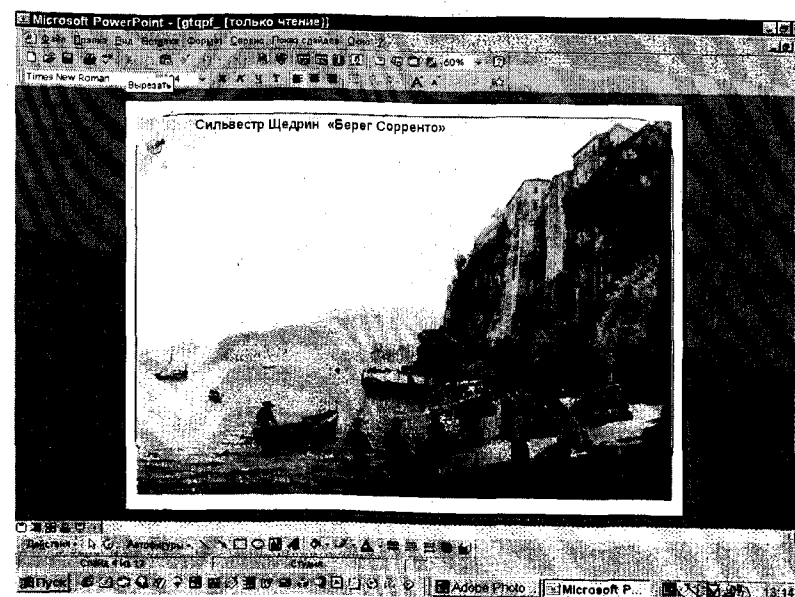


Рис. 29

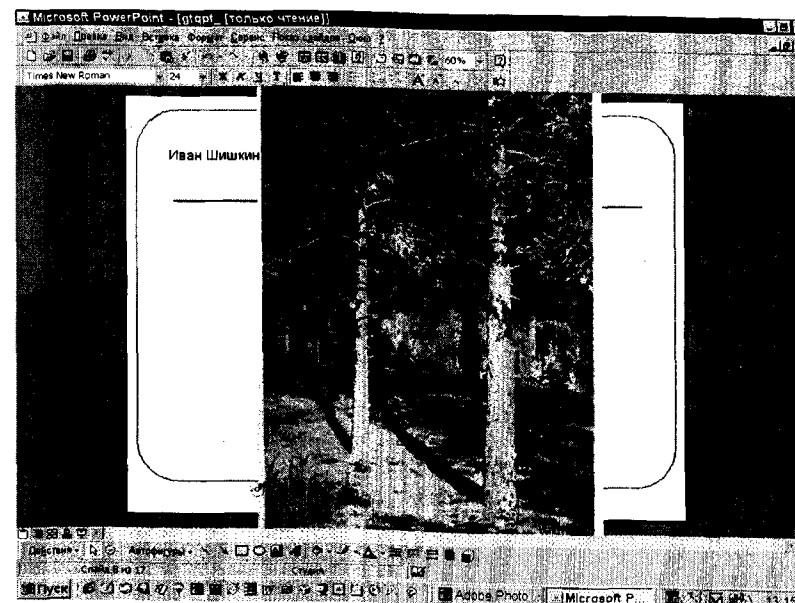


Рис. 30

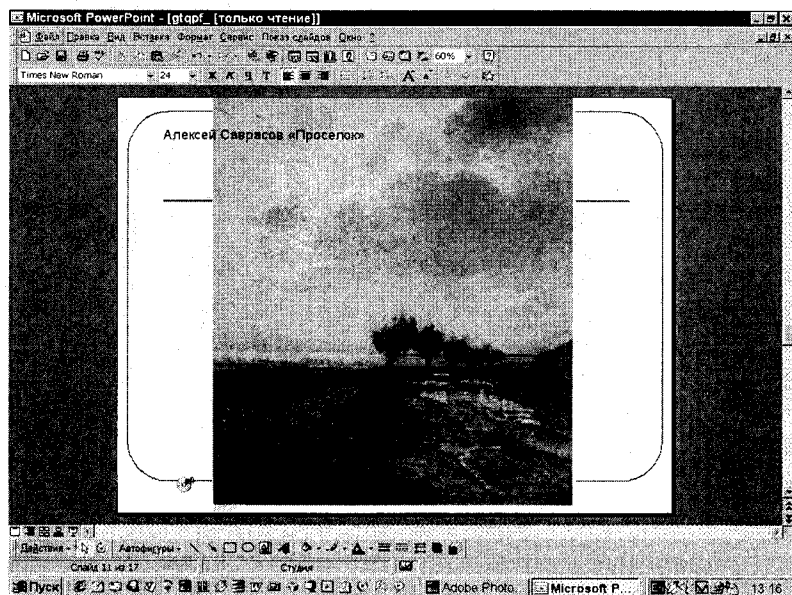


Рис. 31



Рис. 32



Рис. 33



Рис. 34

Программы для анимации позволяют создавать управляемые двух- и трехмерные изображения объектов, комбинировать их для получения анимационных фильмов.

Программы для видео на компьютере — при наличии соответствующего оборудования производят на компьютере монтаж видеofilмов, наложение титров, видеоэффектов и т. д.

Информационно-поисковые системы (ИПС) содержат тексты документов и предоставляют возможности их поиска, выдачи на печать и др. Они могут быть общего назначения (например, все государственное законодательство России) и специализированного (например банковское, таможенное, медицинское и другие законодательства).

Первые информационные системы (ИС), реализованные на электромеханических счетных машинах, появились в 1950-х гг. и в основном предназначались для обработки счетов и расчета зарплаты. В 1960-е гг. информацию из ИС стали применять для отчетности по разнообразным параметрам. В 1970-е и начале 1980-х гг. компьютерные ИС уже широко используют как средство управленческого контроля для принятия решений. К концу 1980-х гг. ИС становятся стратегическим источником информации и широко используются на разных уровнях во многих организациях. Своевременное и быстрое предоставление информации, которое обеспечивают человеко-компьютерные ИС, помогают в любой сфере деятельности для получения рациональных вариантов решения управленческих задач, освобождения работников от рутинной работы за счет ее автоматизации, замены бумажных носителей на магнитные, что снижает объемы бумажных документов, уменьшения затрат на производство продуктов и услуг, а также предоставления уникальных услуг потребителям и др.

По характеру использования информации ИС разделяют на системы информационно-поисковые и информационно-решающие, в которых выделяют информационно-управляющие и советующие.

Управляющие ИС обрабатывают большие объемы информации и выполняют расчетные задачи, в результате вырабатывают информацию, на основе которой человек принимает решения. Это системы бухгалтерского учета (например 1С и др.), оперативного планирования выпуска продукции, создания бизнес-планов — к примеру Project Expert, — и др.

Управляющие ИС входят, как правило, в АСУ: организационные, предназначенные для автоматизации функций управленческого персонала; технологические — для автоматизации функций производственного персонала; интегрированные — для автоматизации

всех функций организации, охватывающие весь цикл работ: от управления до производства продукции.

Советующие ИС вырабатывают информацию, которая принимается к сведению, но не всегда используется для конкретных действий. Это уже интеллектуальные системы, зачастую экспертные, обрабатывающие не данные, а знания.

Среди ИС особо следует выделить геоинформационные системы (ГИС), которые появились в 1960-е гг., но развитие их началось при появлении емких носителей информации и быстродействующих технических средств. Основой ГИС явилась географическая карта, в ней использованы также разные картографические, статистические, аэрокосмические и другие материалы. На основе многослойной топологической модели географических данных в ГИС описываются координаты объектов и их характеристики. Информационной основой баз данных ГИС являются цифровые карты, состоящие из набора тематических слоев данных в цифровой форме. С помощью ГИС получают геоинформацию в пространственной цифровой и нецифровой форме, которая используется во многих сферах деятельности (муниципальном управлении, геологии, строительстве, военном деле, сельском хозяйстве для обустройства территорий, лесоустройстве и т. д.). ГИС подразделяют на тематические (например социально-экономические и др.) и земельные (например кадастровые и др.). Они могут быть общенациональными и региональными, многоцелевыми и специализированными. Одной из главных задач ГИС является поддержание находящейся в ней информации на современном уровне, т. е. задача ее постоянного обновления. Иногда разрабатывают небольшие, локальные так называемые настольные ГИС (например на автомобильные дороги района), но есть примеры создания международных ГИС.

В 1996 г. шесть прибрежных стран создали ГИС «Черное море», которая включает БД по океанографии, геологии, метеорологии, загрязнению, био- и рыбным ресурсам.

Для реализации больших ГИС используют профессиональные инструментально-программные комплексы (например система Intergraph); для небольших — такие программы, как, например, GeoDraw, Geo Graph и др.

Программы бухгалтерские и финансового анализа используют для ведения бухгалтерского учета, подготовки финансовой отчетности и финансового анализа деятельности предприятия. Для предприятий с большим объемом хозяйственной деятельности используют программы складского учета, торговых операций, контроля за выполне-

нием договоров и т. п., которые уже не относятся непосредственно к бухгалтерским. Программы планирования позволяют разрабатывать план работ, требующий для его выполнения много трудовых, материальных и денежных затрат.

Системы автоматизированного проектирования предназначены для проектирования и вычерчивания различных предметов и механизмов на компьютере. Имеются большие, средние и малые системы. Большие включают средства трехмерного моделирования, проектирования процессов обработки деталей, программирования оборудования с числовым программным управлением и т. д.

Экспертные системы ориентированы на решение широкого круга трудно формализуемых задач. В таких системах тесно переплетены технические и творческие элементы. *Экспертная система* — программно-аппаратный комплекс, основой которого является база знаний — система высококачественных специальных знаний о некоторой предметной области.

Экспертные обучающие системы — системы искусственного интеллекта, построенные на основе современных научных знаний о некоторой предметной области для решения различных проблем. В такие системы заложены знания высококомпетентных специалистов в определенных областях. Эти знания хорошо представлены в символической форме, эффективно структурированы и позволяют пользователям целесообразно их использовать. Такие системы, использующие электронные учебники и сеть Интернет, применяют для различных форм дистанционного образования, так как они дают возможность учащимся получать знания, исправлять допущенные ошибки, сдавать зачеты и экзамены путем тестирования (оценки соответствия знаний учащегося экспертной модели тестовых знаний) в устной и письменной форме.

В таких системах, а также компьютерных играх и других досуговых системах широко представлены средства технологии *мультимедиа* — обработки и отображения на компьютере многообразных средств информации: аудио, видео и др.

Технология мультимедиа используется также для подготовки презентаций, стендов и т. п. Мультимедийные проекты занимают большую память, поэтому для их хранения используют магнитные и оптические диски большой емкости (CD-ROM 200—500 Мбайт и выше, DVD и др.).

Технология мультимедиа использует компьютер, воспроизводящий все виды информации, которые вместе называют мультимедиа (multi — много, media — средства информации). Мультимедиа-компьютер должен воспроизводить текст, цветные изображе-

ния, а также аудио и видеоинформацию от разных источников информации: книги, телефона, телевизора, магнитофона, видеокамеры и т. п. Для этого он должен быть снабжен платами, позволяющими ввод-вывод аналоговой информации и ее преобразование в цифровую форму. Для представления аналоговой информации в цифровом виде требуются огромные объемы памяти (например, несколько минут видеофильма занимают десятки Мбайт памяти). Поэтому для этих целей применяют быстродействующие процессоры и оптические компакт-диски. На CD-ROM данные хранятся в виде последовательности цифр, представленных микроскопическими впадинами — питами. С диска их считывает лазерный луч.

Выделяют две группы мультимедийных программ:

- пакеты для обучения и досуга по определенной тематике, энциклопедии по отраслям знаний, электронные учителя в области иностранных языков и т. д. (на CD-ROM от 200 до 500 Мбайт);
- пакеты подготовки видеоматериалов для создания стендов, представлений, демонстрационных дисков (например Multimedia Viewer Kit и др.).

Особый вид мультимедийных проектов — системы «Виртуальная реальность» (virtualis — возможный) — фантастический мир, в котором человек становится действующим лицом. Он, оснащенный специальными средствами (шлемом — дисплеем с двумя небольшими экранами — по одному на каждый глаз, наушниками, датчиком, сообщаемым компьютеру о движении головы, ручным устройством ввода для управления движением (перчатка с датчиками)), видит и активно участвует как бы в реальной жизни, создаваемой информацией, генерируемой компьютером. Система впервые была создана в НАСА (США) при подготовке астронавтов для высадки на Луну, чтобы воссоздать с помощью имитирующих программ те условия, в которых астронавты могут оказаться. Далее система «Виртуальная реальность» получила широкое распространение для создания тренажеров в различных сферах деятельности.

Разработан ряд мультимедийных программ, чтобы будущие врачи могли повышать свою квалификацию без пациентов. На тренажерах можно изучать не только строение внутренних органов, но и происходящие в них процессы.

На тренажерах пилоты отрабатывают действия в полете, автомобилисты — на дорогах, спортсмены учатся плавать, ездить на велосипедах и т. д.

Программы технологии WWW (World Wide Web — Всемирная паутина). Всемирная паутина — это десятки миллионов компьютеров-серверов сети Интернет, содержащих Web-страницы, в которой используется технология гипертекста (текст структурируется с помощью выделения в нем цветом и подчеркиванием слов-ссылок: гиперссылок). Для программирования сайтов используется язык гипертекстовой разметки HTML. С помощью гиперссылок осуществляют переходы как внутри исходного текста, так и на другие документы, находящиеся в компьютерах, подключенных к сети Интернет. Серверы сети, работающие по этой технологии, называются Web-серверы, а документы, созданные по этой технологии, Web-страницы.

Подключиться к Интернету можно через провайдера — лица или организации, имеющих права на выход к серверам глобальной сети. Набрать команды: *Пуск* → *Программы* → *Стандартные* → *Связь* → *Удаленный доступ к сети Провайдер Интернет*. Далее вызвать диалоговое окно *Установка связи*. Ввести идентификатор (login) в поле *Имя пользователя* и пароль (password) в поле *Пароль*. Нажать кнопку *Подключиться*. Должно появиться информационное окно, содержащее характеристики скорости, длительности и др.

Пользователи могут получать информацию с серверов, используя специальные программы просмотра (WWW-браузеры). Например, браузер Internet Explorer (Обозреватель) — специальная программа для навигации по Интернету. Взаимодействие между пользователем (клиентом) и сервером происходит по специальному протоколу HTTP (HyperText Transport Protocol). Пользователь при поиске документа должен указать протокол, адрес сервера, имя директории на сервере и имя файла, в котором находится документ. Например, поиск документа о дешифраторе можно найти в «Большой советской энциклопедии», указав: <http://encycl.yandex.ru/cgi-bin/art.pl>.

Или в учебном пособии «Основы цифровой электроники», указав: <http://www.bdx.ru/wsap/posobie/chapter2/4.htm>.

Запуск Internet Explorer можно выполнить разными способами: из Главного меню, с Панели задач, с Рабочего стола. Открывшееся окно браузера содержит *Меню*, панель *Кнопки*, окно *Адрес* и рабочую область для просмотра Web-страниц. Для просмотра страниц в кодировке Windows необходимо дать команды: *Вид* → *Вид кодировки* → *Кириллица*.

С помощью команд *Сервис* → *Свойства обозревателя* вызывается диалоговое окно *Свойства обозревателя*, в котором следует выбрать *Общие* → *Домашняя страница*. В поле *Адрес* нужно ввести с клавиатуры

адрес — универсальный указатель ресурсов URL (Universal Resource Locator), включающий способ доступа к документу, имя сервера, на котором находится документ, а также путь к файлу (документу). Нажав кнопку *Домой*, можно загрузить начальную страницу этого Web-сайта.

Web-сайты обычно содержат большое количество страниц, поэтому для переходов следует использовать *Панель навигации*, которая содержит названия основных разделов сайта и находится в левой части страницы. Для перехода на другие Web-сайты следует воспользоваться гиперссылками.

Много файловых архивов хранится на так называемых FTP-серверах. Для доступа к ним используется специальный протокол передачи файлов FTP (File Transfer Protocol).

Для определения текущего *IP-адреса* своего компьютера нужно в окне *Сеанс MS DOS* ввести команду *wipnscfg*. На появившейся диалоговой панели Конфигурация IP можно увидеть полную информацию о параметрах текущего подключения к Интернету, включая IP-адрес вашего компьютера.

Для интерактивного общения в Интернете имеется специальное ПО. Общение может быть реализовано в форме разговора, аудиоили видеоконференций. Наиболее распространенной программой является NetMeeting, входящая в Internet Explorer. После ее загрузки нужно в поле *Сервер* из списка выбрать сервер, на котором будет происходить общение. В главном окне будет представлен список участников с некоторыми сведениями о них. Для присоединения к разговору нужно ввести команды *Сервис* → *Разговор* (chat). В поле *Сообщение* окна *Разговор* следует ввести текст сообщений. В поле *Отправить* нужно из списка выбрать получателей сообщения. Для всех участников конференции по командам *Сервис* → *Доска* может быть предоставлена *Доска*, которая позволяет всем участникам создавать и редактировать общий рисунок. Участники могут также коллективно работать с приложениями MS Word (например, по командам *Сервис* → *Общие приложения* → *Текстовый редактор*), Excel и др.

Создание Web-сайтов осуществляют с использованием языка разметки гипертекстовых документов HTML. Технология состоит в том, что в обычный текстовый документ вставляются *управляющие символы (теги)*. В результате получают Web-страницу. Документ может быть создан любым редактором (например Блокнот). Сначала нужно разработать проект сайта, определить сколько и каких страниц будет в нем.

Например, сайт будет содержать титульный лист «Самолеты» и четыре страницы: «Фирмы изготовители», «Комплекующие», «Цены», «Покупатели».

1. Открыть окно текстового редактора Блокнот. Задать вид Web-страницы с помощью тэгов, заключенных в угловые скобки. Пара тэгов называется контейнером. Закрывающий тэг должен содержать прямой слэш (/) перед обозначением.

Web-страница должна содержать заголовок и содержание. Заголовок, содержащий название документа и не отображаемую справочную информацию о странице, нужно заключить в контейнер `<HEAD> </HEAD>`. Название Web-страницы должно быть в контейнере `<TITLE> </TITLE>` и отображаться в строке заголовка программы. Основное содержание страницы нужно поместить в контейнер `<BODY> </BODY>`. Оно может содержать информацию разного вида (текст, рисунок и т. д.). Созданную титульную страницу обязательно сохранить в виде файла под именем `index.htm` или `index.html`. Целесообразно создать для сайта отдельную папку и хранить в ней все созданные страницы.

2. Ввести HTML-код, тогда для титульной страницы в Блокноте будет следующий текст:

```
<HTML>
<HEAD>
<TITLE>Самолеты</TITLE>
</HEAD>
<BODY>
Необходимая информация для Вас —
Самолеты
</BODY>
</HTML>
```

Текст можно отформатировать: заголовок выделить крупным шрифтом (`<H1>` — самый крупный, `<H6>` — самый мелкий), разместить его по центру страницы (атрибут `ALIGN="center"`), выровнять по правой или левой границе (`ALIGN="right"` или `ALIGN="left"`), изменить цвет (например ``). Заголовок следует отделить от содержания горизонтальной линией с помощью тэга `<HR>`.

3. В контейнер `<BODY>` вставить последовательность тэгов

```
<H1 ALIGN="center"
<FONT COLOR="blue">
Необходимая информация для Вас —
Самолеты
</FONT>
</H1>
<HR>
```

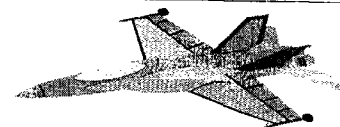
4. Вставить изображение можно, используя тэг `` с атрибутом `SRC="имя файла"`. Например, ``. Атрибут `ALT` позволяет задать поясняющий текст, а `ALIGN` — расположить его с любой заданной стороны от рисунка. Рисунок должен быть в одном из трех форматов: GIF, JPG, PNG. Если формат другой, то его переводят в один из трех с помощью графического редактора (например Photo Editor).

5. В контейнер `<BODY>` вставить тэги:

```
<IMG SRC="sam.gif">
ALT= "Самолет"
ALIGN="left">
```

Результат просмотра нашей титульной страницы — на рис. 35.

Необходимая информация для Вас —
Самолёты



Страницы этого сайта позволяют
вам больше узнать о

**самолётах, их комплектующих,
фирмах изготовителях, ценах,
покупателях**

Рис. 35

Электронная почта наиболее распространенный сервис сети Интернет для обмена информацией. По электронным адресам можно обмениваться письмами-сообщениями, содержащими разные виды информации, включая Web-страницы.

Адрес электронной почты (или E-mail) имеет следующую структуру: *имя пользователя@имя сервера*. Имя задает сам пользователь, а имя сервера связано с сервером, на котором пользователь разместит свой почтовый ящик. Например: адрес *Ivanov@mtu-net.ru* означает, что почтовый ящик Иванова (Ivanov) расположен на почтовом сервере *mtu-net.ru* компании МТУ-Интел, которая работает в России. Для отправки письма по электронной почте, нужно, подключившись к Интернету, передать сообщение на свой почтовый сервер, который отправит его на почтовый сервер получателя в его почтовый ящик. Получатель, выйдя в Интернет, может скачать почту из своего почтового ящика на свой компьютер. Для работы с электронной почтой имеется ряд специальных почтовых программ — например программа Outlook Express.

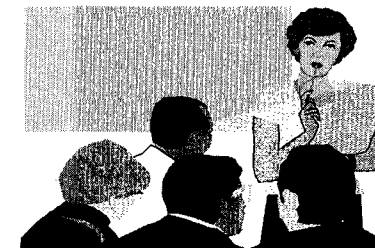
Левая часть окна программы Outlook Express содержит перечень папок с корреспонденцией: входящие, исходящие, отправленные, удаленные, черновики. В папках хранятся объекты разных типов. Правая часть окна разделена пополам. В верхней части показывается содержимое выделенной папки, в нижней — содержимое выделенного сообщения. Чтобы создать почтовое сообщение следует с помощью команд *Сообщение* → *Создать* вывести окно *Пробное сообщение*. В поле *Кому* указать электронный адрес адресата (например *kro@mail.ru*). В поле *Копии* указать адреса получателей копии сообщения. В поле *Тема* указать тему сообщения (например, *Пожелание*). Далее следует ввести текст сообщения в отведенную для него область. В сообщение с помощью команд *Вставка* → *Вложение* файла можно вызвать окно *Вставка вложений* и выбрать файл, созданный другими приложениями графический, звуковой и т. д. Сформировав сообщение и нажав кнопку *Отправить*, можно поместить сообщение в папку *Исходящие*.

Чтобы отправить сообщение адресату, нужно подключиться к Интернету и нажать кнопку *Доставить почту*. Отправленные сообщения будут доставлены на почтовый сервер и одновременно помещены в папку *Отправленные* на вашем компьютере. Почтовый сервер передаст через Интернет ваши сообщения на почтовый сервер получателей, послания окажутся в почтовых ящиках адресатов. Например:

Кому: *kro@mail.ru*
Тема: *Пожелание*

Желаю

здоровья и успехов в учёбе, В.А.



Для получения почты адресат должен соединиться с Интернетом и выполнить операцию доставки почты с почтового сервера на свой компьютер. Сообщения будут размещены в папке *Входящие*.

Для работы с электронной почтой можно использовать Web-технологии. Имеются Web-сайты, которые предлагают бесплатно зарегистрировать свой почтовый ящик (например по адресу: <http://mail.ru>). В этом случае не требуются специальные почтовые программы, работу можно выполнять с помощью любого браузера после загрузки соответствующей Web-страницы, введя свой логин и пароль.

Используя электронную почту, можно участвовать в работе телеконференций. Для этого в окне программы следует выделить папку новостей (например *Конференции*) и нажать кнопку *Группа новостей*. Появится панель с названиями многих конференций. Чтобы найти интересующую тематику, необходимо в поле *Показать группы новостей, содержащие:* задать символьный шаблон (например, *rus* — русскоязычные конференции). Из открывшегося перечня следует выбрать нужные и нажать кнопку *Подписаться*. Содержимое этих конференций будет загружено на компьютер пользователя. Выделяя из папок необходимую, а в ней интересующее сообщение, можно в правом нижнем окне увидеть его содержание. Можно послать на телеконференцию свое сообщение, используя для его создания команды *Сообщение* → *Ответить* на группу новостей.

Контрольные вопросы

1. Какие программы включает внутреннее ПО для БТ?
2. Какие комплексы программ являются внешним ПО?
3. Какие прикладные программы для IBM PC наиболее широко применяются?
4. Что такое экспертная система?
5. Что такое Всемирная паутина и какой язык используется для программирования сайтов?
6. Каковы назначения и основные функции глобальной компьютерной сети Интернет?
7. Каковы назначение и основные функции почтовой программы Outlook Express?
8. Как создать сообщение?
9. Что такое электронная почта с Web-интерфейсом?
10. Что такое работа в телеконференциях?

4.2. Программное обеспечение для автоматизированных систем управления. Стандартные и профессиональные пакеты прикладных программ

Огромный рост объемов информации в сочетании с возможностями ее переработки для целей управления различными процессами дал основу для развития современных АСУ во всех сферах человеческой деятельности. В настоящее время АСУ применяются повсеместно: промышленность, транспорт, системы связи, защита окружающей среды существенно зависят от компьютерных систем управления.

АСУ в нашей стране строились на основе вычислительной техники III-го поколения. Это семейство программно-совместимых вычислительных машин Единой системы (ЕС) ЭВМ 1-й, 2-й и 3-й очереди, созданных в результате сотрудничества социалистических стран. Модели имели быстроедействие до нескольких тысяч операций в секунду. Это также малые машины серии СМ ЭВМ. Их отличали доступность для пользователя, небольшие габариты и стоимость, простота эксплуатации и программирования. Выпускаемые модели имели быстроедействие до 80 тыс. операций в секунду, их можно было подстраивать под конкретные задачи автоматизированного управления и обработки информации. Модели СМ-1, СМ-2 использовали в основном для управления технологическими процессами, производствами и экспериментальными комплексами, для

обработки данных в различных областях, в информационно-поисковых системах, при испытаниях оборудования и т. д. Модели СМ-3, СМ-4 позволяли осуществлять упрощенное программирование, создавать разнообразные вычислительные, управляющие, измерительные, информационные комплексы для решения предметных задач. К этим машинам была возможность подключения микроЭВМ, что обеспечивало построение распределенных систем различного назначения, в которых СМ выполняла функции центральной ЭВМ. Внедрение этих комплексов дало толчок развитию проблемно-ориентированных языков программирования, средств отладки программ, проблемно-ориентированных ППП общего и специального назначения.

К 1990-м гг. при переходе к широкому использованию ПК разрабатываются АСУ, решающие задачи принятия решений для осуществления различных управленческих функциональных процедур.

Применение ЭВМ не только для решения отдельных задач, но и их использование для комплексной автоматизации по переработке информации дало начало стермительному развитию АСУ. Вначале были созданы так называемые АСОД — административные системы обработки данных, т. е. системы для автоматизации банковских операций, бухгалтерского учета, резервирования и оформления билетов и т. п. Основа таких систем — автоматизированные информационные базы — банки данных. Информация таких баз, хранящаяся и обрабатываемая в ЭВМ, и извлекаемая для решения необходимых задач, постоянно используется и обновляется. АСОД по мере развития превратились в АСУ — автоматизированные системы управления разными объектами, которые стали средством эффективной организации информационной среды и управления производством.

В основе всех АСУ лежат следующие принципы:

- наличие различных стандартных вычислительных средств;
- разработка нестандартных аппаратных средств, работающих на базе вычислительной техники;
- использование стандартного и разработка нестандартного ПО для системы;
- использование различных приложений, созданных на основе информационных технологий;
- обработка информации любого вида, путем преобразования ее в цифровую форму и воспроизведения в необходимом виде;
- создание широкой, по мере возможности всеохватывающей, коммуникационной сети, которая может обеспечить постоянную связь между элементами системы.

АС можно разделить по многим классификационным признакам. Но в данном случае выделим системы общего назначения, которые стандартно можно применить в любой сфере деятельности, и системы, профессионально или организационно-ориентированные на определенную сферу деятельности. Первые в своей структуре имеют преимущественно только обеспечивающую часть. Вторые, как правило, имеют и функциональную часть, в которой решаются все вопросы организации производства. Первые в большинстве своем — это автоматизированные системы обработки информации (АСОИ), вторые — автоматизированные системы управления — АСУ (например АСУП — АСУ предприятия), в рамках которой могут функционировать и автоматизированные системы управления технологическими процессами — АСУТП).

Например, структура АСУП может иметь вид, представленный на рис. 36.

АСУ прошли длительный период развития. Обычно выделяют несколько поколений развития систем, связанных с влиянием вне-



Рис. 36

дрения средств ВТ. С 1955—1960 гг. в нашей стране стали применяться средства АСУП. Предприятие обычно приобретало ЭВМ небольшой мощности и с ее помощью решало ряд стоящих перед ним задач. Программирование в основном осуществлялось в машинных кодах, т. к. алгоритмические языки еще только начали появляться.

АСУ 1960-х гг. — это системы обработки на ЭВМ основной производственно-экономической информации: использовали отдельные ЭВМ (или аренду машинного времени) для решения частных задач, важных для данного предприятия. В это время уже разрабатывались вопросы организации ввода исходной информации в ЭВМ, получения промежуточных и окончательных результатов на отдельных фазах автоматизированной обработки данных. Определяется тенденция организации связи различных устройств с ЭВМ. Увеличивается количество задач, решаемых с помощью таких автономных устройств. Если в первой половине 1960-х гг. АСУ создавались на отдельных опытно-показательных предприятиях, то уже во второй половине — стали внедряться первые очереди АСУ в масштабах страны. Разрабатываются автоматизированные системы организационного управления — АСОУ (предприятием — АСУП, отраслью — ОАСУ, АСУ территории), а также АСУТП (для технологических процессов). Технической базой для АСУП и АСУТП в основном являлась ЭВМ «Минск-23». Программирование задач осуществлялось на ассемблерах и таких проблемно-ориентированных языках, как Фортран, Алгол, Кобол.

В 1970-х гг. был значительно увеличен объем создания АСУ. АСУП формируются на базе создаваемых общих и специализированных вычислительных центров (ВЦ), где решались сложные задачи с большим числом переменных. Совершенствуется технология программирования, используются библиотеки типовых проектных решений. Автоматизируются функции оптимального планирования, регулирования и управления на базе экономико-математических методов с использованием статистических данных развития производства. Полностью оформляется автоматизация обработки информации. Разрабатываются ППП системного и функционального назначения.

АСУ начала 1980-х гг. включали не только решение задач управления предприятием, но и АСУТП и создание АРМ — автоматизированных рабочих мест, использующих мини- и микроЭВМ для объекта управления в диалоговом режиме. Разрабатываются автоматизированные системы научных исследований (АСНИ) и системы

автоматизированного проектирования (САПР), включающие комплексы АРМ.

Начинают развиваться интегрированные автоматизированные системы управления (ИАСУ).

В первой половине 1980-х гг. ИАСУ развиваются как по вертикали (автоматизация управления от технологического процесса до отрасли в целом), так и по горизонтали (автоматизация от конструирования изделия до реализации готовой продукции). На ЭВМ III-го поколения создается ПО для решения сложных оптимизационных задач, имитационного моделирования и экспертных систем. Создаются многочисленные ППП. Информационные базы концентрируются в информационно-вычислительных центрах — ИВЦ в виде банков данных, которые используются для решения многих функциональных задач. Банк данных (БНД) — это система, включающая взаимосвязанные данные, хранимые в БД, программные, технические, языковые, организационно-методические средства, предназначенные для обеспечения централизованного накопления и коллективного многоцелевого использования данных. Многие виды учета были объединены в единую интегрированную систему обработки данных.

К 1985 г. был достигнут достаточно высокий уровень интеграции БД, многомашинных и мультипрограммных комплексов. В них использовали крупные ППП и единое методическое обеспечение, позволившее четко выделять функции и процедуры для разных уровней управления. Развитие информационных баз стало основой для наращивания функциональных возможностей АСУ в управлении производством.

В 1990-х гг. переход к ПК и сетевым технологиям открыл широкие возможности интеллектуализации АСУ, путем создания интегрированных технологий, включающих технологии основной деятельности и новые информационные технологии. Непрерывное повышение мощности ПК, периферийных устройств, а также развитие средств связи дало разработчикам ПО больше возможностей для максимально полного удовлетворения запросов конечных пользователей. Программные средства стали создаваться на базе диалоговых интеллектуальных систем.

К настоящему времени АСУ представляют собой гибкие адаптирующиеся интегрированные системы, в том числе ГПС — гибкие производственные системы (ГАП — гибкие автоматизированные производства), с элементами искусственного интеллекта: самонастройки и самообслуживания. Часто ГПС включают САПР построенные по модульному принципу, который позволяет достаточно не-

сложную адаптацию к новым задачам, облегчает поиск ошибок. Современные САПР ориентированы на перспективные технологические процессы, в том числе использование лазерной техники и робототехнических комплексов. САПР, как правило, является одной из подсистем интегрированной АСУ, охватывающей все аспекты функционирования предприятия.

В ГПС используют суперЭВМ, объединенные в сети. Целью при реализации системы является перевод предприятия к безбумажной и безлюдной (малолудной) технологии, в том числе создание заводов-автоматов. Такая технология требует высокоорганизованного информационного, математического и ПО. Информационное обеспечение уже во многих отраслях переведено от независимых локальных массивов нормативно-справочной информации к организации информационных баз и БНД. МО и ПО тоже совершенствуются быстрыми темпами, переходя на новый уровень реализации сложных интеллектуальных задач. Все АСУ сейчас используют новые информационные технологии, средства телекоммуникации и связи, формализации и автоформализации знаний, системные программные средства. Отдельные АСУ для получения доступа к различным БНД подключаются к глобальной сети Интернет. Широко практикуется создание конструкторской и технологической документации на машинных носителях. Часто электронной документацией снабжается ПО. Развиваются языки высокого уровня и средства интеллектуализации.

Программное обеспечение АСУ — это иерархический целевой комплекс (совокупность) программ для реализации ее целей и задач, а также обеспечения работы комплекса технических средств АСУ. ПО должно обеспечить сопряжение вычислительных устройств различного типа, управление вычислительным процессом на каждом уровне системы и в целом, передачу сообщений по каналам связи. ПО тесно связано с МО и составляется на его базе. ПО включает в себя общесистемные и функциональные программные компоненты. Базовое ПО АСУ — программные комплексы, реализующие программную совместимость всех частей АСУ. Это распределенная по уровням программная система, обеспечивающая взаимодействие различных вычислительных средств и управляющая ПО центров коммутации сообщений в системе. Специальные общесистемные программные комплексы, системы проектирования распределенных БД и САПР различных компонентов АСУ служат для осуществления информационной совместимости и единого управления решением функциональных задач.

Можно выделить четыре уровня процессов, протекающих в АСУ, для которых создаются программы разных уровней.

1-й уровень. Сбор данных о ходе производственного процесса от первичных датчиков и преобразователей, использование полученных данных после обработки для прямого программного управления этими процессами.

Для указанных задач создают ПО опроса датчиков и преобразования их по заданным алгоритмам и программы выработки управляющих воздействий на исполнителей.

2-й уровень. Выбор методов обработки результатов измерений и вычислений необходимых параметров.

Для этих задач создают программы обработки результатов измерений и вычислений необходимых параметров.

3-й уровень. Оптимизация производственного процесса и адаптивное управление.

Создают библиотеки стандартных программ оптимального управления (решения оптимизационных задач методами линейного и нелинейного программирования, сетевого планирования, вариационных задач и др.)

4-й уровень. Высший.

Разрабатывают программы для решения задач планирования и управления на базе метода исследования операций в рамках реализации информационно-управляющих систем административно-организационного управления от предприятия и выше.

Основная идея современных методов и средств проектирования ПО экономических информационных систем (ЭИС) заключается в применении инженерного подхода к проектированию ПО. Основная доля трудозатрат при создании ЭИС приходится на прикладное ПО и БД.

В 1970—1980-е гг. была проведена систематизация и стандартизация процессов создания ПО на основе структурного подхода, а в 1990-е гг. начинается переход к сборочному, индустриальному способу создания ПО на основе объектно-ориентированного подхода. В настоящее время в программной инженерии существует два подхода к разработке ПО АС: функционально-модульный, или структурный, и объектно-ориентированный. В первом подходе ПО разрабатывается при разбиении на задачи — декомпозиция по функциям подсистем. При втором подходе структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами.

Развитие информационных технологий определяет постоянное возрастание сложности ПО, поэтому при его разработке возникает множество проблем. Чтобы ПО было эффективно реализовано в проекте, оно должно быть адекватно описано, построены полные и непротиворечивые модели его архитектуры, представляющие собой иерархию подсистем, включающих совокупности структурных элементов системы и связей между ними. Необходимо четко описать поведение элементов в процессе их взаимодействия, разработать средства для визуализации, проектирования и документирования всех элементов системы. ПО должно решать все ее прикладные задачи. Для этих целей широко используются средства программной инженерии (CASE-технологии, см. тему «История развития программирования»).

В АСУ для решения задач подсистемного метода (декомпозиция системы на подсистемы) разрабатывают ППП как совокупность программ, совместимых между собой и обеспечивающих решение задач из некоторой области знаний, называемой предметной областью пакета. При выборе ППП следует учитывать:

- состав реализуемых функций, возможность адаптации и необходимость доработки пакета или его совершенствования;
- надежность пакета с точки зрения защиты данных;
- возможность изменения конфигурации ЭВМ и периферийных устройств, которые предусматривает пакет;
- язык и транслятор, на котором написаны программы, ОС для функционирования ППП, состав стандартных вспомогательных программ;
- возможность применения входных и выходных форм документов, предлагаемых пакетом;
- наличие или возможность получения исходных данных, регламентируемых в пакете;
- наличие и комплектность документации на описание применения ППП.

Разработано большое количество ППП для разных сфер деятельности, например: стандартные офисные пакеты программ, ППП для управленческой деятельности, СУБД, ИПС, системы управления проектами, экспертные системы, системы поддержки решений и др. Большое количество общесистемных и функциональных ППП получили распространение как инструмент автоматизации проектирования АСУ, создания САПР АСУ. Например ППП: АРИУС — реализует функции проектирования архитектуры АСУ; ISDOS, ADS, TAG — реализуют формализацию анализа систем; COD, ИНЕС — проектируют алгоритмы, генерируют программы.

В основе многих из них лежат пакеты программ, разработанные на базе новых информационных технологий (ИТ). ИТ — это автоматизированные способы сбора, обработки, хранения, передачи и использования информации в виде знаний. Они имеют два компонента: машинный и человеческий. По своему существу ИТ — технология интеллектуального доступа человека в вычислительную среду для решения требуемых задач. Проблема совершенствования диалога с ЭВМ — центральная проблема разработки новой ИТ. ИТ лежат в основе современных способов обработки текстовой, числовой и графической информации, создании БД и БнД, систем мультимедиа и виртуальной реальности, искусственного интеллекта, экспертных и обучающих систем, ППП различных научных и инженерных знаний, систем телекоммуникаций, включая Internet.

Совершенствование ИТ должно идти по пути применения естественного языка в информационных системах и прежде всего в системах общения пользователя с ЭВМ.

Имеется много стандартных ППП, реализующих новые ИТ, например:

- обработка текстовой информации — текстовые процессоры (MS Word, Ami Pro, Word Perfect и др.);
- обработка графической информации — графические редакторы (растровые: Paint, Adobe Photoshop, PhotoWorksPlus; векторные: CorelDraw, AldusFreeHand, встроенный в Word и др.);
- распознавание символов (Fine Rieder, Omni Page, Tiger и др.);
- обработка числовой и другой табличной информации — табличные процессоры (Excel, Improv, Quatro Pro и др.);
- верстки документов и подготовки к изданию — настольные издательские системы (PageMaker, QuarkXPress, Corel Ventura, PagePlus и др.);
- демонстрационная графика (Power Point, Harvard Graphics, Director for Windows, M-media Viewer Kit Flash и др.), анимация (3D studio), видео (Adobe Premier);
- обучающие и досуговые системы, системы мультимедиа (энциклопедии по отраслям знаний, тестовые оболочки: г. Москва — Адонис, Аист, Радуга; г. Томск — Фея, MS LikWay и др.);
- САПР (AutoCad, Micro station TurboCad, Компас фирмы «Аскон» и др.);
- организаторы работ (TimeLine, MS Project, Lotus Organizer и др.);

- создание БД и СУБД (dBase разных версий, Paradox, MS Access, MS Oracle и др.);
- поиск информации в БД — информационно-поисковые системы (правовые системы Консультант+, Гарант и др., библиотечные системы);
- обработка финансовой и аналитико-статистической информации (Turbo Tax, 1C, БЭСТ, Stat Graphics и др.), планирования (MS Project Expert);
- экспертные системы по военному делу, геологии, информатике, математике, физике, химии, электронике и др.;
- поиск информации в сети Интернет (Internet Explorer и др.) обмен информацией по электронной почте (Outlook Express и др.).

Разрабатываются интегрированные ППП, которые в свою очередь можно разделить на полносвязанные и объектносвязанные. Полносвязанный ППП (FrameWork, Symphony, LotusWork и др.) — это многофункциональный автономный пакет, где в одно целое объединены функции и возможности различных специальных пакетов, родственных в смысле обработки данных. Объектно-связанный пакет (профессиональный или пользовательский) объединяет специальные пакеты в рамках единой ресурсной базы (MS Office, Borland Office for Windows и др.). Имеется множество функциональных и проблемно-ориентированных ППП, реализующих типовые задачи систем управления: ОАСУ, АСУП, АСУТП, находящихся в фондах Государственного фонда алгоритмов и программ (ГОСФАП), а также отраслевых и республиканских фондах. Прежде чем приступать к разработке нового ПО, всегда следует хорошо изучить уже имеющееся и не тратить впустую денежные и трудовые ресурсы.

В настоящее время разрабатывается ПО для трех основных классов автоматизированных систем: ИАСУ, САПР, ГПС. Большое внимание уделяется созданию ИТ как части общей технологии производства, управления и других сфер деятельности. В каждой технологии должен быть хорошо развитый *интерфейс* — совокупность правил и стандартов технического сопряжения различных устройств, алгоритма их взаимодействия при согласованном выполнении процессов обработки информации.

Дальнейшее развитие средств ПО АСУ идет в направлении разработок для распределенных вычислительных систем, модульных программно-аппаратных интерфейсов, промышленной разработки и поставки программных средств, типизации автоматизируемых функций по всем основным этапам ЖЦ программных продуктов.

Контрольные вопросы

1. На какой вычислительной технике создавались АСУ разных поколений?
2. Какова история развития АСУ в нашей стране?
3. Что такое ГАП?
4. Приведите примеры стандартных ППП, реализующих новые информационные технологии.
5. Каковы тенденции развития АСУ?

Глава 5**ПРАВОВАЯ И ПРОГРАММНАЯ ЗАЩИТА
КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ****5.1. Правовая защита, виды и принципы защиты
компьютерной информации**

Защищаемая информация — это информация, имеющая определенную ценность для субъекта (государства, организации или отдельного человека) и не предназначенная для использования в режиме несанкционированного доступа.

Защита информации — комплекс мер по предотвращению утечки, хищения, утраты, искажения и подделки информации.

Проблема защиты компьютерной информации в настоящее время является одной из первостепенных. Информация должна быть защищена как от несанкционированного доступа, так и от случайных утрат, происходящих по вине пользователей или ненадежной работы аппаратуры.

В Федеральном законе «Об информации, информатизации и защите информации» от 20 февраля 1995 г. № 24-ФЗ определены цели и режим защиты информации, права и обязанности субъектов в этой области, защита права на доступ к информации и т. п. В законе определено, что «защите подлежит любая документированная информация, неправомерное обращение с которой может нанести ущерб ее собственнику, владельцу, пользователю и иному лицу». В то же время этот закон предусматривает, что «отказ в доступе к открытой информации или предоставление пользователю заведомо недостоверной информации могут быть обжалованы в судебном порядке», а также «руководители, другие служащие органов государственной власти, организаций, виновные в незаконном ограничении доступа к информации и нарушении режима защиты информации, несут ответственность в соответствии с уголовным, гражданским за-

конодательством и законодательством об административных правонарушениях».

Закон РФ «О правовой охране программ для электронных вычислительных машин и баз данных» от 23 сентября 1992 г. № 3523-1 устанавливает понятия: программа для ЭВМ, БД, их адаптация, модификация, воспроизведение, распространение, выпуск в свет (опубликование) использование, декомпилирование программы, правообладатель.

Правообладателем может быть «автор, его наследник, а также любое физическое или юридическое лицо, которое обладает исключительными имущественными правами, полученными в силу закона или договора»

Использование программ для ЭВМ и БД пользователями должно осуществляться на основе договора с правообладателем.

По желанию правообладатель может зарегистрировать программу для ЭВМ или БД путем подачи заявки в Российское агентство по правовой охране программ для ЭВМ, БД и топологий интегральных микросхем (Агенство). Правила оформления заявки определяет Агенство. При положительном решении Агенство вносит программу для ЭВМ или базу в Реестр данных программ для ЭВМ или Реестр баз и выдает заявителю свидетельство об официальной регистрации и публикует сведения о регистрации в официальном бюллетене Агентства.

В Законе РФ «Об авторском праве и смежных правах» от 9 июля 1993 г. № 5351-1 (в ред. от 19 июля 1995 г. № 110-ФЗ) программы для ЭВМ определены как объекты авторского права. Правовая охрана распространяется на все виды программ для ЭВМ (такая же, как на произведения литературы), и БД (как сборники) и устанавливается, что не должны ущемляться необоснованным образом законные интересы автора или иного обладателя исключительных прав на программу для ЭВМ или БД. Нарушение авторских прав или прав правообладателя рассматриваются в суде с целью возмещения нанесенного ущерба. Законы предусматривают меры материального возмещения ущерба, включая денежные штрафы и наложение ареста на контрафактную продукцию.

Что касается компьютерных преступлений, то в нашей стране в соответствии со статьями главы «Преступления в сфере компьютерной информации» УК (1997 г.) таковыми являются:

- неправомерный доступ к компьютерной информации;
- создание, использование и распространение вредоносных компьютерных программ;
- нарушение правил эксплуатации компьютерных систем и сетей.

Ответственность за преступления наступает в случае, если «уничтожена, блокирована, модифицирована или скопирована информация, хранящаяся в электронном виде».

Система защиты информации в компьютерной системе — это единый комплекс правовых норм, организационных мер, технических, программных и криптографических средств, обеспечивающих необходимую безопасность защищаемой информации.

Можно рассматривать следующие *виды защиты информации*

Защита данных (data security) — защита данных от несанкционированного (случайного или преднамеренного) изменения, уничтожения или раскрытия.

Защита от персонала (personal security) — обеспечение безопасности, которая защищает людей, работающих в организации, а также компьютерное оборудование от них и от людей, не работающих в организации. Защита от персонала предусматривает процедуры, которые гарантируют, что весь персонал, имеющий доступ к важной информации, имеет требуемые полномочия и соответствующие разрешения.

Защита от копирования (copy protection) — запрет (рассматривается как нарушение закона) на копирование и передачу другим лицам программных продуктов, защищенных авторским правом. Разработчики используют разные методы защиты программ (пароли и др.). Иногда копирование осуществляется с помощью программ-взломок.

Защита каналов связи (communications security — COMSEC) — вид защиты, относящийся к телекоммуникациям и гарантирующих их аутентичность. Защита каналов связи включает контроль (ограничение) доступа, экранирование кабельных линий, использование методов криптографии (шифрования) и средств сетевой защиты — брандмауэров (специальных защитных сооружений).

Защита от излучений (emission security) — совокупность мер защиты, направленных на предотвращение возможности несанкционированного получения информации путем перехвата каналов связи и анализа излучений компьютерной системы.

Физическая защита — обеспечивает безопасность физического компьютерного объекта — здания, компьютерного помещения, самого компьютера, вспомогательного оборудования (НМД, принтеров и др.), носителей информации (дисков, лент, распечаток) и каналов ее передачи (кабелей: волоконно-оптических, на витых парах и др.).

Угрозы могут исходить от стихийных бедствий, нарушения условий эксплуатации, аварий, а также намеренного ущерба. Носителями угроз могут быть необученные или небрежные пользователи,

сотрудники, поставщики и подрядчики, профессиональные преступники разведчики.

Поэтому большое внимание должно уделяться *эксплуатационной защите* путем повышения осведомленности потенциальных жертв относительно возможных преступлений и удержания компьютерного преступника от совершения преступления. Нападения могут происходить из-за снижения уровня обслуживания: исследование содержимого контейнеров для мусора и сбор электронного мусора, перехват информации по каналам связи, запуск «червей» через Интернет, подмена данных, сканирование, отслеживание паролей и т. д.

Для защиты компьютерной информации выработаны следующие принципы.

1. Разумной достаточности. Не нужно стремиться построить абсолютно надежную защиту, так как мощная защита требует больших ресурсов компьютерной системы и ее трудно использовать.

2. Секретности информации о принципах действия защитных механизмов системы. Чем меньше хакерам (взломщикам) известно об этих принципах, тем труднее им организовать успешную атаку.

3. Максимальное ограничение размеров защищаемой сети. Без крайней необходимости не нужно подключать ее к другим сетям, особенно к Интернету.

4. Перед покупкой новых программ следует поискать информацию о них на хакерских серверах Интернет.

5. Охрана помещений, в которых размещаются серверы. Доступ к серверам должен быть только через сеть, не подключая к ним клавиатуру и дисплей.

6. Шифрование всех сообщений, передаваемых по незащищенным каналам связи, и снабжение их цифровой подписью.

7. Сообщения в защищаемую сеть, соединенную с незащищенной сетью, должны проходить через брандмауэр.

8. Ежесуточная проверка журналов событий. В них должны быть зафиксированы только предусмотренные события. Если событий больше, то вероятнее всего, система подвергалась атакам хакеров.

9. Регулярная проверка целостности ПО. Проверяется наличие программных закладок (программ-шпионов), с помощью которых осуществляется перехват, искажение, уборка электронного мусора, наблюдение и компроментация информации.

10. Регистрация всех изменений политики безопасности в обычном бумажном журнале. Это помогает обнаружить присутствие программных закладок, которые внедрил хакер.

11. Пользование только защищенными ОС.

12. Создание ловушек для хакеров (например файл с заманчивым именем).

13. Регулярное тестирование системы с помощью специальных программ, предназначенных для определения степени ее защищенности от хакерских атак.

14. Быть всегда в курсе последних разработок в области компьютерной безопасности и материалов, помещаемых на хакерских серверах Интернет (например asta.lavista.box.sk).

5.2. Программный подход к защите информации.

Антивирусные программы

Как правило, атаки на защиту компьютерных систем происходят на уровнях ОС и сетевого ПО, а затем получают доступ к файлам СУБД с помощью средств ОС.

На уровне ОС это: кража пароля, сканирование жестких дисков, сборка «мусора», превышение полномочий, отказ в обслуживании ОС.

На уровне сетевого ПО это: прослушивание сегмента локальной сети, перехват сообщений на маршрутизаторе, создание ложного маршрутизатора, навязывание сообщений, отказ в обслуживании сети. Поэтому сеть должна быть ограничена в размерах, по возможности изолирована от внешнего мира, сообщения должны шифроваться и иметь электронные цифровые подписи.

Обеспечение правильного функционирования компьютерной системы может быть достигнуто, если в системе отсутствуют программы, мешающие ее работе. К таким вредоносным программам, от которых следует избавляться, относятся компьютерные вирусы и программы-шпионы: программные закладки, троянские программы, клавиатурные шпионы, фильтры, заместители.

Классификацию компьютерных вирусов можно провести по степени опасности, среде обитания, способу заражения и алгоритму функционирования.

По степени опасности компьютерные вирусы могут быть *безвредными* (уменьшают только свободную память из-за размножения), *неопасные* (добавляют разные эффекты), *опасные* (приводят к сбоям в работе компьютера), *очень опасные* (приводят к потере программ и данных, форматированию жестких дисков и т. д.)

По среде обитания компьютерные вирусы делят на файловые (внедряются в исполняемые файлы и активизируются при их запус-

ке), загрузочные (заносятся в загрузочный сектор диска, при загрузке ОС внедряются в оперативную память), макровирусы (заражают файлы документов Word и Excel) и сетевые (любые вирусы, распространяющиеся по сети, часто они внедряются в компьютеры через зараженные файлы с серверов файловых архивов, по электронной почте, как вложенный в сообщение файл, или через Всемирную паутину, в виде активных элементов — программ на языке Java или Visual Basic).

По способу заражения среды обитания компьютерные вирусы могут быть резидентными, если после активации они попадают из среды обитания в оперативную память ЭВМ и остаются там, и нерезидентными, т. е. попадают в оперативную память ЭВМ только на время активации.

По алгоритму функционирования вирусы делят на:

- изменяющие среду обитания (файлы, сектора) при распространении («студенческие» — низкой квалификации, «стелс» — невидимки, маскирующие свое присутствие под программы ОС, «полиморфные» — не имеющие постоянных опознавательных групп и использующие шифрование тела вируса);
- не изменяющие среду обитания («спутники», создающие копии исполняемых файлов, «черви» и др.).

Любой вирус независимо от класса имеет три блока: заражения, маскировки и выполнения деструктивных действий.

Защита от вирусов состоит в следующем:

- не следует исполнять файлы, полученные от неизвестного источника;
- все вновь полученные программы должны быть проверены антивирусными программами;
- ОС не следует загружать с гибких дисков;
- в BIOS компьютера должна быть установлена защита загрузочного сектора от изменений;
- если при открытии Word и Excel имеется сообщение о наличии потенциальных вирусов, то компьютер должен быть отключен, или вылечен от вируса специальной программой;
- запрещается получать по сети на компьютер активные элементы.

Теперь рассмотрим программы-шпионы.

Программные закладки могут вносить произвольные искажения в коды программ, переносить информацию из одних областей памяти в другие, искажать выводимую на внешние устройства информацию. Программные закладки могут быть программно-аппаратные,

загрузочные, драйверные, прикладные, исполняемые, замаскированные и др. Если программная закладка попадает в оперативную память компьютера, то при активизации программ она начинает либо копировать информацию, либо изменять алгоритмы функционирования системных, прикладных и служебных программ, либо навязывать определенные режимы работы.

Модели воздействия программных закладок: перехват, искажение, «уборка мусора», наблюдение и компрометация.

При перехвате закладка внедряется в постоянное запоминающее устройство (ПЗУ) компьютера, системное или прикладное ПО и сохраняет выбранную информацию в скрытой области памяти локальной или удаленной системы.

Закладка может исказить, изменить информацию. Выделяют статическое (одноразовое) и динамическое (изменяющее параметры процессов с помощью заранее активизированных закладок) искажение. Часто закладки на искажение используют хакеры для изменения цифровых подписей.

При работе пользователь часто оставляет копии документов в корзине и дальше не заботится об их защите. Эта информация может быть восстановлена и использована в корыстных целях. Поэтому следует заботиться об «уборке мусора».

Закладки наблюдения встраиваются в сетевое ПО и следят за обработкой информации в системе, могут также устанавливать и удалять другие программные закладки. Закладки компрометации обеспечивают доступ к информации, перехваченной другими закладками.

Защита от программных закладок состоит в том, чтобы не допустить внедрения их в систему, или выявить и удалить. Но самым универсальным средством защиты от закладок является создание изолированного компьютера, т. е. компьютера прошедшего проверку на их наличие:

- устанавливается BIOS, не содержащая закладок;
- проверяется ОС;
- BIOS и ОС проверяются каждый сеанс работы;
- не допускается работа с посторонними программами;
- осуществляется постоянный контроль, путем использования специального драйвера контроля;
- используются средства перепрограммирования ПЗУ компьютера.

Обнаруживают программные закладки по качественным и визуальным признакам, а также средствами тестирования и диагностики.

Троянская программа — программа, которая является частью другой (пользовательской) программы, но тайне от пользователя выполняющая вредные действия. Троянская программа, по существу, — это разновидность программной закладки. Предназначены они в основном для нанесения ущерба компьютерной системе или сбора информации о пользовательских паролях, регистрационных номерах программ, сведениях о банковских счетах и других конфиденциальных данных. Широко известны такие троянские программы, как PC CYBORG, AOLGOLD, Back Orifice, Net Bus, SubSeven и др. Надо очень осторожно относиться к предлагаемым бесплатным программам, т. к. они могут оказаться троянцами.

Для защиты от троянских программ используют программы согласования объектов — файлов и каталогов. При этом следят, изменились ли они после последней проверки. Проверяют атрибуты файлов, их размеры, контрольные суммы элементов файлов, в том числе используя функцию одностороннего хэширования* для отслеживания изменений вносимых злоумышленником (например утилита TripWire для проверки ОС UNIX). Для борьбы с такими троянскими программами в ОС Windows, как Back Orifice, Net Bus, SubSeven, можно использовать антивирусные программы Norton Antivirus 2000 фирмы Symantec, The Cleaner фирмы MooSoft Development, пакет eSafe Protect компании Aladin Knowledge Systems и др.

Клавиатурные шпионы — программные закладки, нацеленные на перехват паролей пользователей ОС и определение легальных полномочий пользователей и прав доступа к компьютерным ресурсам. Имеется три типа клавиатурных шпионов: *имитаторы* (имитирующие приглашение зарегистрироваться для входа в систему), *фильтры* (забирают информацию, вводимую с клавиатуры), *заместители* (полностью или частично подменяют собой программные модули ОС, отвечающие за проверку пользователей). Против клавиатурных шпионов вводят специальные административные меры и используют программно-аппаратные средства защиты.

Для защиты от имитаторов (перехвата паролей) создают специальные программы, отвечающие за аутентификацию (проверку) пользователей. Средства аутентификации не должны быть доступны другим процессам и прикладным программам. В ОС Windows NT за это отвечает системный процесс Win Logon, имеющий свой рабочий стол (совокупность окон) аутентификации. В начальном окне этого

* Hash function — хэш-функция (функция хэширования) — позволяет получить на выходе результат фиксированного размера при подаче на вход данных произвольного объема.

стола предлагается нажать клавиши <Ctrl>+<Alt>+. Далее идет переключение в регистрационное окно, в котором в соответствующие поля пользователь вводит свое имя и пароль, проверяющие процесс. Для защиты от фильтров не разрешается переключение клавиатуры во время ввода пароля. Конфигурацию цепочки программных модулей и доступ к файлам может определять только системный администратор. Для защиты от заместителей администраторы системы должны соблюдать политику безопасности и только сами конфигурировать подсистему аутентификации.

Пароль является основным защитным рубежом против злоумышленников в компьютерной сети. Чтобы пользователь мог работать с ОС, он должен зарегистрироваться, задать имя и пароль. Например, окно входа в Windows 98 представлено на рис. 37.

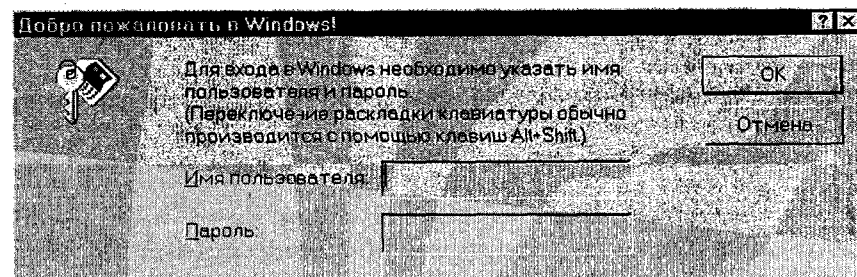


Рис. 37

Парольную защиту устанавливают в следующем порядке:

- 2 раза щелкнуть мышью по пиктограмме *Мой компьютер* (My Computer);
- 2 раза щелкнуть мышью по пиктограмме *Панель управления* (Control Panel);
- 2 раза щелкнуть мышью по пиктограмме *Пароли* (Passwords);
- в появившемся диалоговом окне *Свойства паролей* щелкнуть на опции *Сменить пароль Windows* (Change Windows Passwords);
- ввести новый пароль в поле *Новый пароль* (New Passwords);
- поле *Старый пароль* (Old Passwords) не заполняется;
- ввести также в поле *Подтверждение пароля* (Confirm New Passwords) новый пароль и нажать ОК.

Система сообщит, что пароль успешно установлен (сменен).

Для MS Windows 2000 окно входа показано на рис. 38.

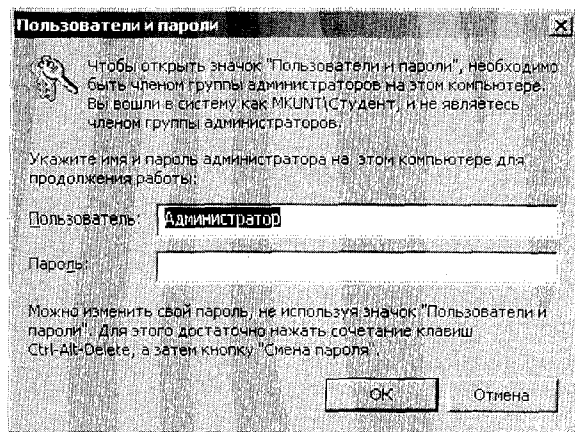


Рис. 38

Если системный файл с именами и паролями зашифрован, то именно он чаще всего атакуется. Есть парольные взломщики — специальные программы, которые служат для взлома ОС. Поэтому особое внимание следует уделять защите паролей и особенно парольной защите ОС и конфиденциальной информации. Каждый пользователь обязан перед началом работы зарегистрироваться, т. е. сообщить свое имя (для идентификации) и пароль (для подтверждения правильности идентификации), который сравнивается с имеющимся в ОС. Доступ к файлам с зашифрованными именами и паролями запрещен даже системным администраторам.

Для защиты паролей используют следующие методы:

- ограничение физического доступа к компьютерной сети;
- затенение: в файле passwd зашифрованные пароли заменяются служебными словами;
- отсутствие пароля в словарях взломщиков.

Шифрование — процесс преобразования открытого текста с целью сделать непонятным его смысл для посторонних. Различают шифры замены и шифры перестановки.

Шифры замены производят замену каждой буквы открытого текста на символ зашифрованного текста. Получатель расшифровывает его путем обратной замены. Имеется несколько разновидностей шифров замены: простая, или одноалфавитная, многоалфавитная, блочная и др.

В **шифрах перестановки** меняется порядок следования букв. Но эти шифры легко взламываются, поэтому предпочтение отдается шифрам замены.

Шифрование появилось задолго до появления компьютеров. А в 1920-е гг. были изобретены устройства, позволяющие автоматизировать процесс шифрования и расшифрования. Большинство из них состояло из клавиатуры для ввода открытого текста и набора роторов — специальных вращающихся колес, каждое из которых реализовывало простую замену (например А на Ф и т. д.). Во время Второй мировой войны для засекречивания переписки использовалась, например, немецкая роторная машина «Энигма».

Для создания шифров, которые нелегко раскрыть, разрабатывают специальные алгоритмы. Отыскать методы их расшифровки очень сложно.

Трудность атаки (взлома, расшифровки) характеризуется сложностью данных, временем вычислений, объемом памяти. Она имеет вид экспоненциальной функции. Например, если сложность атаки 2^{64} , то это значит, что для взлома шифра требуется выполнить 2^{64} операций.

Максимальное время, требуемое для взлома пароля, можно вычислить по формуле

$$T = \frac{1}{S} \sum_{i=1}^L N^i,$$

где N — число символов в наборе; L — предельная длина пароля; S — количество проверок в секунду.

Эта формула показывает, что для взлома парольной защиты требуется много времени, так как надо перебрать огромное количество ситуаций. Чем больше символов в пароле, тем большее число комбинаций следует выполнить. Взломщики применяют специальные словари — заранее сформированный список слов, наиболее часто употребляемых в качестве паролей. Чтобы затруднить им возможность подбора пароля с помощью таких словарей, нужно выбирать несколько оригинальных и не коротких паролей.

Например, для аутентификации пользователей ОС Windows NT следует обратиться в БД SAM, где хранится зашифрованная информация о пользовательских именах и паролях. Длина пароля в Windows NT ограничена 14 символами. Каждый пароль представлен в базе двумя 16-байтовыми последовательностями, полученными разными методами. При входе в систему введенный пользователем пароль сначала хэшируется, преобразуясь в более короткий, который затем шифруется, а далее сравнивается с 16-байтовой последовательностью, записанной в базе SAM. При совпадении паролей разрешается вход в систему. Для защиты этой базы есть специаль-

ная утилита SYSKEY, которая позволяет включить режим дополнительного шифрования информации о паролях базы SAM. Там имеется 128-битовый ключ шифрования паролей — Password Encryption Key (PEK).

Утилита Диспетчер пользователей (User Manager) помогает для стойкости паролей в ОС Windows NT с помощью диалогового окна задать пароль длиной не менее 8 символов, периодически его обновляя. Пароль не должен содержать имени пользователя, а должен включать символы — не менее трех наборов: прописные и строчные буквы, цифры и специальные символы.

Системный администратор обязан следить, чтобы в БД, где хранятся пароли, не было несанкционированного доступа, а также ограничить доступ к контроллерам доменов и с помощью специальных программно-аппаратных средств установить пароли BIOS на включение компьютеров и изменение настроек BIOS, не допускать загрузку с гибких дисков, следить за хранением дискет аварийного восстановления и т. д. Для защиты паролей применяют и специальные утилиты.

Для шифрования паролей применяются стойкие криптографические алгоритмы. Криптография — наука, изучающая сохранение содержания сообщений втайне. Область науки о вскрытии шифров называется криптоанализом. Под криптосистемой понимается алгоритм шифрования, множество всевозможных ключей, открытых и зашифрованных текстов. Криптография помогает сохранить содержание сообщения втайне, обеспечить аутентификацию отправителя сообщения, его целостность и неоспоримость.

Криптографический алгоритм, называемый шифром или алгоритмом шифрования, представляет собой математическую функцию, используемую для шифрования и расшифрования. Любой криптографический алгоритм может быть реализован в виде программы.

Для шифрования используют ключ K (Key), который должен выбираться среди значений, принадлежащих множеству, называемым ключевым пространством. Например, обозначим:

P — открытый текст, обязательно двоичные данные (файл, битовое изображение, оцифрованный звук и т. д.);

C — шифротекст, тоже двоичные данные;

E — функция шифрования;

D — функция расшифрования.

$$E(P) = C;$$

$$D(C) = P.$$

После шифрования преобразованный текст может передаваться по каналам сети или быть сохранен в памяти компьютера.

Смысл любого криптографического преобразования открытого текста в том, чтобы потом его можно было восстановить в первоначальном виде:

$$D(E(P)) = P.$$

Функции E и D зависят от ключа K , можно записать:

$$E_K(P) = C;$$

$$D_K(C) = P;$$

$$D_K(E_K(P)) = P.$$

Если ключ шифрования отличен от ключа расшифрования, то формулы примут следующий вид:

$$E_{K1}(P) = C;$$

$$D_{K2}(C) = P;$$

$$D_{K2}(E_{K1}(P)) = P.$$

Почти во всех ОС имеются встроенные средства шифрования файла. Обычно они предназначены для шифрования отдельных файлов, а с ключами работает пользователь. Категорически запрещается хранить ключи на диске вместе с файлами, зашифрованными с их помощью, а незашифрованные копии файлов необходимо удалять сразу после шифрования. До шифрования файл следует сжать.

Например: рассмотрим, как спрятать один шифротекст в другом.

Пусть один студент шифрует текст $P \oplus K = C$. Второй студент, имея ключ K , может расшифровать сообщение первого: $C \oplus K = P$. Символ \oplus в математике означает сложение по модулю 2 и представляет собой стандартную операцию над битами: $0 \oplus 0 = 0$; $0 \oplus 1 = 1$; $1 \oplus 0 = 1$; $1 \oplus 1 = 0$.

С помощью сложения по модулю 2 можно выполнять многоалфавитную замену, прибавляя к битам ключа соответствующие биты открытого текста. Такой алгоритм является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, а шифрование и расшифрование выполняется одной и той же программой.

Криптоанализ заключается в получении доступа к открытому тексту зашифрованного сообщения. Попытка проведения криптоа-

нализа — это атака. Успешная атака называется взломом, или вскрытием.

Надежность (стойкость) алгоритма шифрования с использованием ключей достигается за счет тщательного выбора и хранения ключей в строжайшем секрете. Стойкость шифра зависит от сложности его взлома криптоаналитиком. Если криптоаналитик не может восстановить исходный открытый текст, то криптографический алгоритм называется безусловно стойким. Стойкие ключи, как правило, имеют длину от 45 до 75 бит. Но иногда для защиты цифровых подписей в сложных системах рекомендуют 512 бит и более. Например, если длина ключа 56 бит, то можно иметь 2^{56} ключей и выбирать любой из этого ключевого пространства. Хороший ключ — это случайный битовый вектор. Для хранения ключей используют магнитную карточку или пластиковый ключ с ПЗУ, или интеллектуальную карту ключей.

Иногда для защиты информации используют одноразовые блокноты — алгоритмы, которые невозможно вскрыть, так как это очень длинная последовательность случайных букв, записанная на листах, которые скреплены в блокнот. Отправитель сообщения использует каждую букву из блокнота, чтобы зашифровать ровно одну букву открытого текста. Шифрование состоит в сложении буквы открытого текста и буквы одноразового блокнота по модулю N , где N — количество букв в алфавите. После шифрования отправитель уничтожает использованный одноразовый блокнот. Чтобы отправить новое сообщение, ему следует изготовить новый. Получатель, владеющий копией одноразового блокнота, получает открытый текст путем сложения букв шифротекста и букв из копии одноразового блокнота. Затем он эту копию уничтожает.

В настоящее время имеется ряд стандартов шифрования.

В СССР, а затем в России — это ГОСТ 28147—89. Симметричный алгоритм шифрования.

В США — это DES (Data Encryption Standard). Симметричный алгоритм шифрования. Государственный стандарт США. Алгоритм использует 56-битный ключ, а его ключевое пространство содержит 2^{56} ключей.

Стандарт ANSI X9.17 (на основе генерации 64-битных ключей) разработан Американским национальным институтом стандартов ANSI с использованием DES-алгоритма.

Стандарт RSA (авторы: Rivest, Shamir, Adleman). Алгоритм шифрования с открытым ключом.

Антивирусные программы — программы, предназначенные для поиска вирусов и лечения от них зараженных файлов. В России ан-

тивирусными исследованиями занимаются компании «Лаборатория Касперского» и «СалД».

Можно выделить следующие виды антивирусных программ: полифаги, ревизоры, блокировщики.

Наиболее эффективными являются универсальные антивирусные программы — *полифаги* (например AntiViral Toolkit Pro, AVP Monitor и др.), которые проверяют файлы, сектора дисков и оперативную память. Поиск вируса заключается в отыскании постоянной последовательности программного кода (метод маски), присущей определенному вирусу. Может анализироваться также последовательность команд («эвристическое сканирование»). К недостаткам этих программ относят небольшую скорость поиска вирусов и большие размеры антивирусных баз.

Широко распространены программы-«доктора» AIDSTEST.EXE, DRWEB.EXE.

По номеру версии программы AIDSTEST.EXE можно установить, какое количество вирусов она «лечит».

Программе DRWEB.EXE известно немного вирусов, но она обладает способностью находить файлы подозрительные на наличие незнакомого ей вируса.

Запустив файл DRWEB.EXE, пользователь видит перед собой оболочку с меню, расположенным в верхней строке экрана. Для проверки выбираются пункты *Тест* → *Тестирование* и указывается путь для выполнения теста. После нажатия <Enter> открывается Окно тестирования. С помощью пункта *Настройка* задаются общие установки тестирования, уровень эвристики, высота экрана, цветовая схема и т. д., а также действия при обнаружении зараженных файлов.

После проверки выдается Окно отчета о выполненном тестировании.

Программы-ревизоры (например, Adinf и др.) подсчитывают контрольные суммы файлов, сохраняют эти суммы, длины файлов, даты их изменений в антивирусной БД. При новом запуске данные базы сверяются с реальными, расхождения обнаруживаются как вирус. Недостаток этих программ — невозможность обнаружить вирус в новых программах, так как в антивирусной базе о них нет необходимых данных.

Программы-блокировщики перехватывают и сообщают пользователю о сомнительных ситуациях, например при установке ОС — о записи в загрузочный сектор дисков. Антивирусные блокировщики часто «зашивают» в *Setup* компьютера. Эти программы блокируют вирус, не давая ему размножаться.

Для защиты от вирусов следует использовать самые последние версии антивирусных программ. Следует изучать каталоги известных вирусов, в которых указывается их имя, длина, заражаемые файлы, место внедрения в файл, метод заражения, способ внедрения в ОП, вызываемые сбои.

Для «лечения» компьютера от вирусов следует:

- иметь самые последние версии антивирусных программ;
- при подозрении на вирус в компьютере проинформировать об этом системного администратора;
- отключить компьютер от сети;
- запустить антивирусную программу;
- при обнаружении загрузочного вируса проверить абсолютно все диски;
- проверить оперативную память на наличие вируса и удалить его;
- обнаружить зараженные файлы, удалить или «вылечить» их.

В России разработаны такие программные системы защиты ПЭВМ от несанкционированного доступа, как «Снег-1.0», «Кобра», «Страж 1.1» и др., а также аппаратно-программные средства (системы) защиты: «Аккорд-4», «Редут» и др., которые уже сертифицированы.

Контрольные вопросы

1. Какова правовая основа защиты информации?
2. Каковы виды и принципы защиты информации?
3. Что такое компьютерные вирусы и какова их классификация?
4. Что такое программы-шпионы и каковы их виды?
5. Что такое пароль и как устанавливают парольную защиту?
6. Что такое шифрование и криптографический анализ?
7. Каковы виды антивирусных программ и их характеристики?

Глава 6

СТАНДАРТИЗАЦИЯ И ЛИЦЕНЗИРОВАНИЕ ПРОГРАММНЫХ ПРОДУКТОВ

В международном стандарте ISO/IEC 12207:1995 ПП (программный продукт) определен как «набор компьютерных программ, процедур и, возможно, связанной с ними документации и данных».

Программные продукты — программы, которые могут предназначаться как для собственного потребления, так и для продажи. Во втором случае нужно учитывать множество условий, определяемых разными потребителями, в том числе создание документации, удобной и понятной для пользователей, а также тестировать программы с разными наборами данных.

Сложность, многогранность и универсальность программных продуктов, массовость их применения потребовали стандартизации как самих программ — программных средств (ПС), так и процессов их разработки.

6.1. Правовые акты стандартизации и сертификации программных продуктов

Прежде всего, программист должен хорошо знать действующие в стране законы, регламентирующие области работ, с которыми он соприкасается. Ему должны быть хорошо известны основные положения следующих федеральных законов:

1. «О стандартизации» от 10 июня 1993 г. № 5154-1.
2. «О сертификации продукции и услуг» от 27 апреля 1993 г. № 5151-1 (в ред. от 27 декабря 1995 г. № 211-ФЗ; от 2 марта 1998 г. № 30-ФЗ; от 31 июля 1998 г. № 154-ФЗ);
3. «Об информации, информатизации и защите информации» от 20 февраля 1995 г. № 24-ФЗ;
4. «О правовой охране программ для электронных вычислительных машин и баз данных» от 23 сентября 1992 г. № 3523-1;
5. «Об участии в международном информационном обмене» от 4 июля 1996 г. № 85-ФЗ;

6. «Об авторском праве и смежных правах» от 9 июля 1993 г. № 5351-1 (в ред. от 19 июля 1995 г. № 110-ФЗ).

В соответствии с первым федеральным законом «стандартизация — это деятельность по установлению норм, правил и характеристик (далее — требования) в целях обеспечения:

- безопасности продукции, работ и услуг для окружающей среды, жизни, здоровья и имущества;
- технической и информационной совместимости, а также взаимозаменяемости продукции;
- качества продукции, работ и услуг в соответствии с уровнем развития науки, техники и технологии;
- единства измерений;
- экономии всех видов ресурсов;
- безопасности хозяйственных объектов с учетом риска возникновения природных и техногенных катастроф и других чрезвычайных ситуаций;
- обороноспособности и мобилизационной готовности страны.

К нормативным актам по стандартизации относятся:

- «...государственные стандарты РФ; применяемые в установленном порядке, международные (региональные) стандарты;
- правила, нормы и рекомендации по стандартизации;
- общероссийские классификаторы технико-экономической информации;
- стандарты отраслей, предприятий, научно-технических, инженерных обществ и других общественных объединений».

В соответствии со вторым законом «сертификация продукции (далее — сертификация) — процедура подтверждения соответствия, посредством которой независимая от изготовителя (продавца, исполнителя) и потребителя (покупателя) организация удостоверяет в письменной форме, что продукция соответствует установленным требованиям».

Сертификация осуществляется в целях:

- «содействия потребителям в компетентном выборе продукции;
- защиты потребителя от недобросовестности изготовителя (продавца, исполнителя);
- контроля безопасности продукции для окружающей среды, жизни, здоровья и имущества;
- подтверждения показателей качества продукции, заявленных изготовителем.

Сертификация может иметь обязательный и добровольный характер».

Обязательная сертификация проводится в случаях, предусмотренных законодательными актами РФ. Организация и проведение работ по обязательной сертификации возложены на Госстандарт России и другие федеральные органы исполнительной власти РФ.

Добровольная сертификация проводится по инициативе заявителей (изготовителей, продавцов, исполнителей) для того, чтобы подтвердить соответствие продукции требованиям стандартов, технических условий и других документов, определяемых заявителем. Проводится она на условиях договора между заявителем и органом по сертификации.

В третьем законе определены основные принципы разработки, производства, сертификации и лицензирования информационных систем, технологий и средств их обеспечения (а следовательно и ПО).

В четвертом законе установлены многие основные понятия и определения создания и использования программ. Например, *программа для ЭВМ* определяется как «объективная форма представления совокупности данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств с целью получения определенного результата, включая подготовительные материалы, полученные в ходе разработки программы для ЭВМ, и порождаемые ею аудиовизуальные отображения». *База данных* определяется как «объективная форма представления и организации совокупности данных, систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны с помощью ЭВМ».

В пятом законе определен правовой режим участия в международном информационном обмене и установлены правила контроля и ответственности при осуществлении международного информационного обмена.

В шестом законе даны основные понятия авторского права, правила использования и защиты авторских произведений, в том числе программ для ЭВМ. В ст. 25 этого закона «Свободное воспроизведение программ для ЭВМ и баз данных. Декомпилирование программ для ЭВМ» описаны условия, при которых можно использовать программу без получения разрешения автора или иного обладателя прав на нее. Это только лица, *правомерно владеющие* экземпляром программы для ЭВМ или БД. Ни при каких обстоятельствах не должны быть ущемлены законные интересы автора или иного обладателя исключительных прав на программу для ЭВМ или БД.

В нашей стране действует Единая система программной документации (ЕСПД), представляющая собой комплекс взаимосвязанных государственных стандартов в области программирования. Эти стандарты регламентируют все виды программ и программной доку-

ментации, процессы их разработки, оформления и обращения (сопровождение, тиражирование и др.).

В состав ЕСПД входят:

- основополагающие и организационно-методические стандарты;
- стандарты, определяющие формы и содержание программных документов, применяемых при обработке данных;
- стандарты, обеспечивающие автоматизацию разработки программных документов.

В 1990-е гг. издан также сборник межгосударственных стандартов «Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы». Эти стандарты устанавливают виды, наименование, комплектность и обозначение документов, в том числе для ПО, разрабатываемых на стадиях создания АС.

6.2. Стандартизация, сертификация и лицензирование программных продуктов

Документирование разработки, сопровождения и эксплуатации программ выполняют в соответствии с группой стандартов ГОСТ 19.XXX—XX.

Предусматривается построение обозначений стандартов в следующем порядке:

- номер 19 — класс стандартов ЕСПД;
- после 19 ставится точка и далее одна цифра — код классификационной группы стандартов, определенной группы стандартов;
- далее идет двузначное число — порядковый номер стандарта в группе;
- через тире за ним ставится двузначное число — год регистрации стандарта.

Так, например, обозначение ГОСТ 19.001—77 расшифровывается как Государственный стандарт «Единая система программной документации. Общие положения», зарегистрированный в 1977 г.

ПС, разработанное для продажи, называют ПИ — программным изделием. ПИ — это программа на носителе данных, являющаяся продуктом промышленного производства (ГОСТ 19.004—80).

ПИ имеет свой «цикл жизни» — период времени от момента возникновения идеи (или необходимости) по разработке программы до момента окончания ее эксплуатации. Между этими моментами

ЖЦ программы проходит три стадии: разработку, использование, сопровождение.

На все отдельные компоненты ПИ также имеются стандарты, что облегчает работу программистам в части написания внешних функциональных спецификаций, позволяя им сосредоточиться на разработке наиболее сложных (творческих) моментах программ.

Ниже приведены государственные стандарты на компоненты стандартизируемой продукции.

ГОСТ 19.101—77, введен с 1981 г. Виды программ и программных документов.

ГОСТ 19.102—77. Стадии разработки.

ГОСТ 19.103—77. Обозначения программ и программных продуктов.

ГОСТ 19.104—78, введен с 1981 г. Основные надписи.

ГОСТ 19.105—78, введен с 1981 г. Общие требования к программным документам.

ГОСТ 19.106—78, введен с 1981 г. Требования к программным документам, выполненным печатным способом.

ГОСТ 19.201—78, введен с 1981 г. ТЗ. Требования к содержанию и оформлению.

ГОСТ 19.202—78. Спецификация.

ГОСТ 19.301—79, введен с 1983 г. Программа и методика испытаний.

ГОСТ 19.401—78, введен с 1983 г. Текст программы.

ГОСТ 19.402—78, введен с 1981 г. Описание программы.

ГОСТ 19.403—79. Ведомость держателей подлинников.

ГОСТ 19.404—79. Пояснительная записка.

ГОСТ 19.501—78. Формуляр.

ГОСТ 19.502—78, введен с 1981 г. Описание применения.

ГОСТ 19.503—79, введен с 1981 г. Руководство системного программиста.

ГОСТ 19.504—79, введен с 1981 г. Руководство программиста.

ГОСТ 19.505—79, введен с 1981 г. Руководство оператора.

ГОСТ 19.506—79, введен с 1981 г. Описание языка.

ГОСТ 19.507—79, введен с 1981 г. Ведомость эксплуатационных документов.

ГОСТ 19.508—79. Руководство по техническому обслуживанию.

ГОСТ 19.601—78. Общие правила дублирования, учета и хранения.

ГОСТ 19.602—78. Правила дублирования, учета и хранения программных документов, выполненным печатным способом.

ГОСТ 19.603—78, введен с 1981 г. Общие правила внесения изменений.

ГОСТ 19.604—78, введен с 1981 г. Правила внесения изменений в программные документы, выполненные печатным способом.

ГОСТ 19.701—90, введен с 1992 г. Схемы алгоритмов, программ, данных и систем.

Стандартизация помогла унифицировать и автоматизировать процесс создания программ на базе инструментальных и программных средств, создать системы автоматизированного программирования с использованием инструментальных и программных средств. К настоящему времени АС позволяют унифицированно выполнять следующие процессы:

- анализ задачи, разбиение ее на подзадачи;
- анализ структур данных;
- запись требований к программе и разработку ее общей структуры;
- выделение модулей, написание их спецификаций, определение интерфейса между ними;
- вычерчивание блок-схем алгоритмов;
- непосредственное программирование (кодирование);
- отладку и тестирование;
- анализ качества и количества затраченного труда на разработку ПИ.

Стандартизация улучшает контроль и регламентацию труда программистов, поэтому иногда она встречает у них психологический барьер. Нужно разъяснять в таких случаях всю полезность и необходимость стандартизации и в качестве квалифицированных программистов принимать участие в разработке стандартов для внесения в них интересных и полезных новшеств.

В теме 1.3 уже рассматривались виды и этапы создания программ в соответствии с ГОСТ 19.781—90. Программное обеспечение систем обработки информации. Этот стандарт по многим позициям повторяет ГОСТ 19.101—77, ГОСТ 19.102—77 и ГОСТ 19.201—78, в которых установлены виды программ: компонент — программа и комплекс — программа, а также стадии разработки программной документации: техническое задание, эскизный проект, рабочий проект, внедрение.

В ГОСТ ЕСПД 19.101—77 дана информация о видах стандартных программных и эксплуатационных документов. Приводится их перечень и описание. Некоторые из программных документов рассмотрены в теме 1.3. Опишем их подробнее.

Программные документы и их содержание:

- спецификация — перечень и назначение всех файлов ПИ, включая файлы документации;
- ведомость держателей подлинников — список предприятий, хранящих подлинники программных документов, составляется только для сложных ПИ;
- текст программы — запись кодов программы и комментарии к ним;
- описание программы — информация о логической структуре и функционировании программы;
- программа и методика испытаний — перечень и описание требований, которые должны быть проверены в ходе испытания программы, методы контроля;
- техническое задание — документ, в котором излагаются назначение и область применения программы, требования к ПИ, стадии и сроки разработки, виды испытаний;
- пояснительная записка — обоснование принятых и примененных технических и технико-экономических решений, схемы и описание алгоритмов, общее описание работы ПИ.

К программным документам отнесены также документы, обеспечивающие функционирование и эксплуатацию программ — эксплуатационные документы:

- ведомость эксплуатационных документов — содержит список эксплуатационных документов на ПИ, к которым относятся формуляр, описание применения, руководство системного программиста, руководство программиста, руководство оператора, описание языка, руководство по техническому обслуживанию;
- формуляр — содержит основные характеристики ПИ, состав и сведения об эксплуатации программы;
- описание применения — содержит информацию о назначении и области применения ПИ, ограничениях при применении, классе и методах решаемых задач, конфигурации технических средств;
- руководство системного программиста — содержит сведения для проверки, настройки и функционирования программы при конкретном применении;
- руководство программиста — содержит сведения для эксплуатации ПИ;
- руководство оператора — содержит подробную информацию для пользователя, обеспечивающую его общение с ЭВМ в процессе выполнения ПИ;
- описание языка — содержит синтаксис и семантику языка;

- руководство по техническому обслуживанию — содержит сведения для применения тестовых и диагностических программ при обслуживании технических средств.

В этом же стандарте приводятся коды программных документов и указывается, что спецификация и текст программы — обязательные документы, разрабатываемые на стадии рабочего проектирования программы-компонента (спецификация — и для программы-комплекса). Необходимость составления остальных документов устанавливается при разработке и утверждении технического задания (ТЗ).

Кроме формуляра и ведомости допускается объединять отдельные виды эксплуатационных документов. Например, часто разрабатывают документ, называемый «Руководство пользователя», в который включают различные сведения из руководства системного программиста, руководства программиста и оператора. «Руководство пользователя» должно учитывать все требования инструкций, необходимых пользователю и содержать, как правило, общие сведения о ПИ, описание установки и запуска, подробные инструкции по работе, т. е. описание режимов работы, форматов ввода-вывода информации, различных настроек и другой необходимой информации для пользователя.

Большую роль для унификации при программировании играет стандарт ГОСТ 19.701—90 (ИСО 5807—85) «Схемы алгоритмов, программ, данных и систем», где приведены условные обозначения в схемах алгоритмов, программ, данных и систем, устанавливаются правила выполнения схем для решения различных задач. В этом стандарте описаны последовательности описания схем:

- данных (отображают путь данных, этапы обработки, носители);
- программы (отображают последовательность операций в программе);
- работы системы (отображают управление операциями и поток данных в системе);
- взаимодействия программ (отображают путь активаций программ и взаимодействий с соответствующими данными);
- ресурсов системы (отображают конфигурацию блоков: данных и обрабатывающих).

Созданию пакета документов на разработку, изготовление, сопровождение и эксплуатацию программ придается очень большое значение. В зависимости от документирования программа может быть и комплексом, и компонентом. Программный документ записывают на таких носителях, как МД, МЛ и др. Документы в зависи-

мости от способа их изготовления и использования могут быть подлинником, дубликатом или копией (ГОСТ 2.102—68):

Подлинник — документ, заверенный установленными подписями и считающийся первичным. Он может быть изготовлен на любом материале и с него могут изготавливаться копии и дубликаты.

Дубликат — документ, скопированный с подлинника, полностью идентичен подлиннику.

Копия — документ, скопированный с подлинника или дубликата, используется при сопровождении и эксплуатации программ.

Программный продукт, как и любой другой, должен быть сертифицирован, т. е. пройти проверку и иметь документ, удостоверяющий его соответствие требованиям стандартов и других нормативных документов. Имеется два сертификационных документа: сертификат соответствия и знак соответствия.

Сертификат соответствия — это документ, выданный по правилам системы сертификации для подтверждения соответствия сертифицированной продукции установленным требованиям.

Знак соответствия — это зарегистрированный в установленном порядке знак, которым по правилам определенной системы сертификации подтверждается соответствие маркированной им продукции установленным требованиям.

Система сертификации — совокупность участников сертификации, которые проводят сертификацию продукции по устанавливаемым в этой системе определенным правилам в соответствии с законом. Система сертификации создается федеральными органами исполнительной власти.

Сертификация включает следующие этапы:

- 1) подача заявки на сертификацию;
- 2) рассмотрение и принятие решения по заявке;
- 3) проведение необходимых проверок (анализ документов, испытания, проверка и т. п.);
- 4) анализ полученных результатов и принятие решения о возможности выдачи сертификата соответствия;
- 5) выдача сертификата и лицензии (разрешения) на применение знака соответствия;
- 6) инспекционный контроль за сертифицированным объектом в соответствии со схемой сертификации.

Схема сертификации — определенная совокупность действий, официально принимаемая в качестве доказательства соответствия продукции заданным требованиям.

На рис. 39 приведена форма заявки на проведение сертификации продукции, а на рис. 40 — форма сертификата соответствия.

ФОРМА ЗАЯВКИ НА ПРОВЕДЕНИЕ СЕРТИФИКАЦИИ ПРОДУКЦИИ

наименование органа по сертификации, адрес

ЗАЯВКА НА ПРОВЕДЕНИЕ СЕРТИФИКАЦИИ ПРОДУКЦИИ В СИСТЕМЕ СЕРТИФИКАЦИИ

наименование системы

1. _____
наименование предприятия-изготовителя, продавца (далее — заявитель),

код ОКП-О

Юридический адрес _____

Телефон _____ Факс _____ Телекс _____

в лице _____

Ф., И., О. руководителя

заявляет, что _____

наименование вида продукции, код ОКП

_____ Выпускается серийно или партия (каждое изделие при единичном производстве)

_____, выпускаемая* по _____

наименование и реквизиты

_____, соответствует требованиям _____
документации изготовителя (ТУ, стандарт) _____
наименование и обозначение стандартов

и просит провести сертификацию данной продукции на соответствие требованиям указанных стандартов по схеме _____

номер схемы сертификации

2. Заявитель обязуется:
- выполнять все условия сертификации;
 - обеспечивать стабильность сертифицированных характеристик продукции, маркированной знаком соответствия;
 - оплатить все расходы по проведению сертификации.

3. Дополнительные сведения _____

Руководитель предприятия _____

подпись, инициалы, фамилия

Главный бухгалтер _____

подпись, инициалы, фамилия

Печать _____

Дата

*Если заявителем является продавец, то после слова "выпускаемая" записывается "изготовителем"

наименование изготовителя

Рис. 39

СЕРТИФИКАТ СООТВЕТСТВИЯ

СИСТЕМА СЕРТИФИКАЦИИ ГОСТ Р

ГОССТАНДАРТ РОССИИ



(1) _____

№ _____

СЕРТИФИКАТ СООТВЕТСТВИЯ

(2) № _____

(3) Действителен до " __ " _____ г.

**НАСТОЯЩИЙ СЕРТИФИКАТ УДОСТОВЕРЯЕТ, ЧТО ДОЛЖНЫМ
ОБРАЗОМ ИДЕНТИФИЦИРОВАННАЯ ПРОДУКЦИЯ**

(4) _____

наименование

(5)

код К—ОКП

тип, вид, марка

(6)

код ТН ВЭД

размер партии

СООТВЕТСТВУЕТ ТРЕБОВАНИЯМ НОРМАТИВНЫХ ДОКУМЕНТОВ

(7) _____

ИЗГОТОВИТЕЛЬ (ПРОДАВЕЦ) (8) _____

наименование,

адрес,

(9) _____

документы (сертификаты, аттестаты и т. п.) о стабильности производства

М. П.

Рис. 40

Сертификат выдан на основании: (10)

Наименование испытательной лаборатории	№ протокола испыта- ний, дата утвержде- ния	Регистрационный № испытательной лаборатории в Гос- реестре
(11)	(12)	(13)

Изготовитель (продавец) обязан обеспечить соответствие реали-
зуемой продукции требованиям нормативных документов, на соответ-
ствие которым она была сертифицирована, испытанному образцу:

(14)

Место нанесения знака соответствия

(15)

В случае невыполнения условий, лежащих в основе выдачи сертифи-
ката, действие его отменяется органом по сертификации, выдавшим
сертификат, или Госстандартом России.

М. П.

Руководитель органа, выдавшего сертификат

(16)

подпись

инициалы, фамилия

Зарегистрирован в Государственном реестре

(17) " ____ " _____ 199 ____ г.

Рис. 40. Продолжение

Литература

1. Анин Б. Ю. Защита компьютерной информации. СПб.: БХВ-Петербург, 2000.
2. Бешенков С. А., Григорьев С. Г., Гейн А. Г. Информатика и информационные технологии. Екатеринбург, 1995.
3. Благодатских В. А., Енгибарян М. А., Ковалевская Е. В. и др. Экономика, разработка и использование программного обеспечения ЭВМ: Учебник. М.: Финансы и статистика, 1995.
4. Бутко А. И. Введение в программирование: Учебное пособие. Омск, 1998.
5. Годин В. В., Корнеев И. К. Информационное обеспечение управленческой деятельности. М.: Мастерство и Высшая школа, 2001.
6. Единая система программной документации (ЕСПД). М.: Изд-во стандартов, 2000.
7. Жуков А. Изучаем Delphi. СПб., М., Харьков, Минск: Питер, 2000.
8. Законы РФ: «О стандартизации» от 10.06.1993; «О сертификации продукции и услуг» от 27.04. 1993; «Об информации, информатизации и защите информации»; «О правовой охране для электронных вычислительных машин и баз данных» от 23.09.1992 и др.
9. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. М.: Изд-во стандартов, 2000.
10. Керниган Б., Ритчи Д. Язык программирования Си. М.: Финансы и статистика, 1992.
11. Кершан Б., Новембер А., Стоун Дж. Основы компьютерной грамотности / Пер. с англ. под ред. Е. К. Масловского. М.: Мир, 1989.
12. Крылова Г. Д. Основы стандартизации, сертификации и метрологии: Учебник. М.: ЮНИТИ, 2000.

13. Лапчик М. П., Семакин И. Г., Хеннер Е. К. Методика преподавания информатики. М.: АCADEMIA, 2001.
14. Ляхович В. Ф. Руководство к решению задач по основам информатики и вычислительной техники. М.: Высшая школа, 1994.
15. Могилев А.В., Пак Н.И., Хеннер Е.К. Информатика. М.: Academia, 2000.
16. Немнюгин С. Turbo Pascal: Учебник. СПб.: Питер, 2000.
17. Немнюгин С. Turbo Pascal: Практикум. СПб.: Питер, 2000.
18. Нестеренко А. В. ЭВМ и профессия программиста. М., 1990.
19. Пратт Т. Языки программирования: разработка и реализация. М., 1979.
20. Научные основы организации, управления и построения АСУ. М.: Высшая школа, 1990.
21. Симонович С., Евсеев Г. Занимательное программирование на C++. М.: АСТ-ПРЕСС КНИГА, 2003.
22. Страуструп Б. Язык программирования C++. М.: Радио и связь, 1991.
23. Угринович Н. Информатика и информационные технологии. М.: Лаб. Базовых Знаний, 2000.
24. Флоренсов А. Н. Введение в программирование. Семантический подход: Учебное пособие. Омск, 1998.
25. Шаров В. Г. Введение в программирование: Учебное пособие. Ярославль, 1991.
26. Объектно-ориентированное программирование.
<http://www.object.newmail.ru/index.html>
27. Объектно-ориентированные языки программирования.
http://www.citforum.ru/win/programming/ooop_rsis/glava1_2.s.html

Глоссарий

Автокод — набор псевдокоманд для решения специализированных задач.

Автоматизированная информационная система (АИС) — человеко-машинная система (взаимосвязанная совокупность средств, методов и персонала), обеспечивающая сбор, хранение, обработку, поиск и выдачу необходимой потребителю информации.

Автоматизированная система управления (АСУ) — человеко-машинная система, реализующая автоматизированный сбор и переработку информации, необходимой для принятия решений по управлению объектом. АСУ создают для оптимального управления в различных сферах деятельности.

Адресация — указание на размещение данных в памяти ЭВМ и извлечение их.

Адресация относительная — термин, используемый в электронных таблицах и означающий, что при копировании формул, адреса ячеек меняются в соответствии с указываемой ячейкой, в которой идет вычисление по формуле.

Адресация абсолютная — термин, используемый в электронных таблицах и означающий, что при копировании формул адреса ячеек не меняются в соответствии с указываемой ячейкой, в которой идет вычисление по формуле.

Алгол (Algol) — один из первых языков программирования высокого уровня, который дал начало целой серии алголоподобных языков.

Алгоритм — совокупность правил и указаний, сформулированных на некотором языке и направленных на достижение определенной цели.

Алгоритмический язык — язык описания последовательности действий для решения задачи.

Алгоритм линейной структуры — последовательность команд следующих одна за другой в естественном порядке.

Алгоритм разветвляющейся структуры — последовательность команд, порядок следования которых определяется в зависимости от результатов проверки некоторых условий.

Алгоритм циклической структуры — последовательность команд, порядок следования которых обеспечивает получение результата многократным их повторением.

Алгоритмический язык — это система обозначений и правил для единообразной и точной записи алгоритмов и исполнения их. Язык записи алгоритма должен быть понятным для человека.

Алфавит языка программирования — фиксированный набор символов, допускаемых для составления текста программы на этом языке.

Антивирусная программа — программа, которая отслеживает распространение всех видов вирусов (специальных вредных саморазмножающихся программ) на компьютере и по возможности лечит зараженный объект, удаляет его или запрещает доступ к нему.

Ассемблер — машиноориентированный язык программирования, созданный на основе мнемонических кодов, использующих символьную адресацию. Код ассемблера записывается в виде мнемоник (сокращенных слов английского языка).

База данных (БД) — именованная совокупность структурированных, организованных данных, отображающая состояние объектов и их отношений в определенной предметной области.

База знаний — именованная совокупность организованных данных и знаний в определенной предметной области и логические правила манипулирования ими для получения необходимых, в том числе новых, знаний.

Базовые операции с файлом — запись в файл (помещение данных в файл), чтение из файла (извлечение хранимых в файле данных), открытие файла (получение доступа к содержимому файла), закрытие файла (окончание доступа к содержимому файла).

Банк данных (БнД) — система специально организованных данных, программных, языковых, организационных и технических средств, предназначенных для централизованного накопления и коллективного многоцелевого использования данных.

Библиотека стандартных программ — собрание (комплексы, совокупность), как правило, небольших, но часто используемых программ (процедур, функций и т. д.), к которым обращается программист для включения необходимых из них в свою программу.

Блокировщики — антивирусные программы, перехватывающие и сообщаящие пользователю о сомнительных ситуациях, например, при установке ОС — о записи в загрузочный сектор дисков. Эти программы блокируют вирус, не давая ему размножаться.

Булева алгебра — алгебра логики, явилась инструментом разработки сложных логических схем, в том числе ЭВМ.

Ветвление — базовый алгоритмический элемент, позволяющий строить структуры по условиям.

Виртуальная реальность — создаваемый на экране компьютера фантастический мир, подобный образам реального мира и процессам в нем происходящим.

Внешнее (общесистемное) ПО ВТ — библиотеки типовых программ (ввода, контроля, сортировки, корректировки, дублирования, обработки единиц информации, поиска и вывода информации), программы решения конкретных задач, системная диспетчерская программа.

Внутреннее ПО ВТ — базовые, эксплуатационные программы (тестовые и диагностические), проверяющие исправность оборудования ЭВМ, систем программирования и операционных систем. Внутреннее ПО тесно связано со структурой ЭВМ и реализует возможности, заложенные в аппаратуре.

Выражение бывает двух видов: арифметическое, логическое. *Арифметическое выражение* соответствует принятому в математике. Например, арифметическое выражение $(a + b)\sin x + c^2 \operatorname{tg} y$ будет записано, например на Паскале, как $(a+b)*\sin x + c^2*\operatorname{tg} y$. *Логическое выражение* образуется с помощью знаков логических операций из объектов (констант, переменных и т. д.) бу-

квенного типа. В логическое выражение может входить *отношение* — одно из фундаментальных понятий языков программирования. Например, записи: $a < b$, $c >= d$, $x^2 + y^2 < a^2$ — отношения, а запись $(a < b) \text{ or } (c = n)$ — логическое выражение.

Глобальное имя переменной — имя переменной, которое может использоваться во всех подпрограммах, и описывается перед всеми подпрограммами.

Данные — информация, представленная в виде, позволяющем хранить, передавать или обрабатывать ее с помощью технических средств.

Дельфи (Delphi) — одна из популярных современных систем для разработки программ, т. е. автоматизированная система программирования, работающая на платформе ОС Windows. Delphi — диалоговая система, т. е. способная выполнять оперативный обмен сообщениями между пользователем и ЭВМ. В Delphi можно программировать разные проекты, в том числе анимационные, писать программы, предназначенные для сети Интернет, организовывать ссылки на различные сайты Всемирной паутины и т. д.

Диалог — оперативный обмен сообщениями между пользователем и ЭВМ. В системе диалог может быть организован как *примитивный*, т. е. собеседники не очень обращают внимание на ответы друг друга, *одноступенчатый*, т. е. один вопрос — один ответ, *двухступенчатый*, т. е. сначала следует один вопрос, а после ответа на него — второй, в зависимости от ответа на первый, *многоступенчатый*, аналогичен двухступенчатому, но с разным количеством вопросов и ответов.

Дисциплина «Программирование» — самостоятельная научная область знаний, занимающаяся методами разработки программного обеспечения. Она также включает методы инженерных расчетов, математические методы и методы управления.

Драйверы — важный класс системных программ, расширяющих возможности операционной системы, позволяя работать с тем или иным внешним устройством.

Единая система программной документации (ЕСПД) — комплекс взаимоувязанных государственных стандартов в области программирования. Эти стандарты регламентируют все виды программ и программной документации, процессы их разработ-

ки, оформления и обращения (сопровождение, тиражирование и др.).

Жизненный цикл (ЖЦ) ПО — период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации.

Загрузчик — программа, которая переносит объектную программу из внешней памяти в оперативную память ЭВМ.

Защита информации — комплекс мер по предотвращению утечки, хищения, утраты, искажения и подделки информации.

Защищаемая информация — информация, имеющая определенную ценность для субъекта (государства, организации или отдельного человека) и не предназначенная для использования в режиме несанкционированного доступа.

Знак соответствия — зарегистрированный в установленном порядке знак, которым по правилам определенной системы сертификации подтверждается соответствие маркированной им продукции установленным требованиям.

Идентификатор — последовательность букв и цифр, начинающаяся с буквы. В основном идентификаторы употребляются в качестве *имен* каких-либо объектов (констант, переменных, типов, структур, программ и т. д.), что позволяет идентифицировать эти объекты.

Иерархическая модель БД — модель описания объектов, входящих в предметную область как совокупности иерархически (по уровням подчинения) взаимосвязанных объектов и их атрибутов (свойств). Отношения построены по принципу «один ко многим».

Инкапсуляция — объединение в единое целое данных и алгоритмов обработки этих данных.

Интерпретатор — программа, анализирующая каждую инструкцию (каждый шаг) транслируемой программы отдельно, после чего осуществляющая шаг за шагом (с проверкой) автоматический перевод программы в машинные коды и выполняющая их.

Информация — сведения о людях, предметах, фактах, событиях и процессах, независимо от формы их представления.

Информационные технологии — машинизированные способы хранения, передачи, обработки и использования информации в виде знаний.

Исполнитель алгоритма — человек, механическое устройство, робот, компьютер и др., способные понимать и выполнять команды алгоритма.

Исполняемый код (рабочая программа) — откомпилированная и отредактированная программа, запускаемая на исполнение.

Инструментальная среда программирования — инструментальная программная оболочка, облегчающая составление и отладку программ на ПЭВМ.

Каталог — перечень файлов с указанием их места хранения на дисках. Каталог при создании называется именем и предназначен для хранения информации о файлах, а не самих данных. Передвигаться по каталогам можно с помощью простейших команд ОС.

Киберпространство — совокупность всех систем компьютерных коммуникаций и потоков информации, циркулирующих в мировых сетях.

Кобол (Cobol) — один из первых языков программирования высокого уровня, предназначенный для решения экономических и управленческих задач.

Кодирование — перевод команд алгоритма на команды строго формализованного языка, языка программирования.

Компилятор — программа, преобразующая в ходе непрерывного процесса весь файл исходного текста в машинный код, после чего осуществляет его выполнение.

Компьютерные вирусы — саморазмножающиеся (самокопирующиеся) программы, внедряющиеся в файлы, загрузочные сектора дисков и документы и наносящие им ущерб или уничтожающие их.

Конвертор — программа, принимающая на входе файл одного формата, преобразует его, и выдает файл в другом (необходимом) формате. Как правило, входят в систему подготовки текстов.

Криптоанализ — область криптографии о вскрытии шифров.

Криптография — наука, изучающая сохранение содержания сообщений втайне.

Локальное имя переменной — имя переменной, которое может использоваться только в подпрограмме, после заголовка и перед телом которой оно описывается.

Машинные коды — символы или их совокупности, которые понимает процессор ЭВМ. Если программа написана в машинных кодах, то она сразу может исполняться ЭВМ. Машинный код, как правило, записывается в шестнадцатичной системе счисления, двучисленными числами или обозначениями (например, 33, F6 и т. д.). Каждому коду отводится один байт.

Метаязык — язык для описания языка. В качестве метаязыка используют либо металингвистические формулы Бэкуса — Наура (БНФ — специальная система определений для языков программирования), либо синтаксические диаграммы (графические изображения допустимых структур определяемых понятий).

Моделирующие программы — межмашинные программы, позволяющие проверить (отладить) объектную программу без ЭВМ.

Модульное программирование — разработка программ с использованием декомпозиции (разбиения) задачи на подзадачи (модули), связанные только с входными и выходными данными. Модульное программирование позволяет разрабатывать части программ одного проекта на разных языках программирования, а затем с помощью компоновочных средств объединять их в один загрузочный модуль.

Набор приложений к ОС — программы (пакет программ), позволяющие обрабатывать информацию разных видов.

Наследование — свойство объектов порождать своих потомков.

Объект — предмет или образ реального мира, который обладает набором характерных свойств и их значений. Свойства объектов, сохраняющие свое значение, являются *постоянными*, а изменяющие свое значение — *переменными*.

Объектно-ориентированное программирование (ООП) — программирование, в котором как основная логическая единица рассматривается объект, который содержит данные и правила

(методы) их обработки. ООП основано на принципах инкапсуляции, наследования и полиморфизма.

Объектный код — откомпилированная программа, исходный код записанный в машинных кодах.

Оператор — предложение языка программирования, основная единица программы, команда, задающая действия компьютеру, выполняющая определенные действия над данными.

Операционная оболочка — удобный и наглядный способ общения пользователя с компьютером, созданный системными программами.

Операционная система — комплекс специальных управляющих программ для функционирования ЭВМ и организации решения задач на ЭВМ. ОС управляет всеми узлами ЭВМ и внешними устройствами и обеспечивает требуемый режим обработки данных.

Отладка — процесс, позволяющий с помощью специальных программ (отладчиков) находить выделяемые ошибки в транслируемой программе. Отладочные программы облегчают отладку объектной программы на ЭВМ.

Пакеты прикладных программ (ППП) — структурно-сложные системы программ, предназначенные для решения задач определенного класса. ППП, как правило, состоят из программ общего и функционального назначения. Первые реализуют типовые режимы работы ВТ, вторые обеспечивают обработку данных.

Пароль — ряд символов, обеспечивающих защиту информации от несанкционированного доступа. Является основным защитным средством против злоумышленников в компьютерной сети.

Паскаль — первый язык высокого уровня, в котором четко реализованы принципы структурного программирования.

Подпрограмма — именованная группа программных действий, которая может быть вызвана из других мест программы. Подпрограмма один раз описывается, но может многократно использоваться.

Подпрограмма-процедура — подпрограмма, которая не возвращает ни одного значения или возвращает более одного значения. Тип результата не задается.

Подпрограмма-функция — подпрограмма, которая обязательно возвращает результат, который затем используется в программе. Тип результата должен быть задан.

Полиморфизм — свойство родственных объектов решать схожие по смыслу проблемы разными способами.

Полифаги — наиболее эффективные универсальные антивирусные программы (например AntiViral Toolkit Pro и др.), которые проверяют файлы, сектора дисков и оперативную память. Отыскивают постоянную последовательность программного кода, присущую определенному вирусу.

Пользовательское ПО — программы, которые служат для выполнения конкретных задач во всех сферах человеческой деятельности.

Прикладное программирование — разработка прикладных программ.

Проблемно-ориентированный язык программирования — язык, предназначенный для написания программ в определенной области знаний, ориентированный на решение задач по определенным проблемам. Например, Фортран был разработан для математических и технических расчетов, Кобол — для действий с экономической информацией.

Программа (исходный код) — упорядоченный список команд (инструкций, включающих операторы и параметры), записанных на языке программирования, для управления компьютером или решения задач.

Программа-комплекс — программа, состоящая из двух и более компонентов и (или) комплексов, выполняющих взаимосвязанные функции, и применяемая самостоятельно или в составе другого комплекса.

Программа-компонент — программа, рассматриваемая как единое целое, выполняющая законченную функцию, и применяемая самостоятельно или в составе комплекса.

Программа линейной структуры — последовательность выполняемых друг за другом операторов, а также необходимых служебных слов и комментариев, записанных на языке программирования.

Программа разветвляющейся структуры — последовательность предложений, записанных на языке программирования, предусматривающих выбор одного из нескольких последовательностей операторов (ветвей) в зависимости от некоторых условий.

Программа циклической структуры — последовательность предложений, записанных на языке программирования, порядок следования которых позволяет многократно вычислять группу операторов при одновременном изменении одного или нескольких параметров.

Программирование — процесс описания последовательности действий решения задачи средствами конкретного языка программирования и оформление результатов описания в виде программы. Если раньше программирование сводилось к написанию кода, то теперь оно включает сбор и анализ требований, проектирование, кодирование, дизайн, документирование, реализацию, управление изменениями, тестирование, а также менеджмент (управленческий механизм). И сам процесс получил название более соответствующее его сути — разработка программного обеспечения.

Программист — человек со специальной подготовкой (специалист) в области программирования, умеющий применять полученные знания для создания программ.

Программное обеспечение АСУ — иерархический целевой комплекс программ для реализации целей и задач АСУ, а также обеспечения работы ее технических средств.

Программное обеспечение для ВТ и АС — совокупность программ и необходимой документации для обеспечения работы комплекса технических средств и реализации целей и задач АС различного направления. ПО тесно связано с математическим обеспечением (МО) и составляет на его базе.

Программное управление компьютером — обеспечение функционирования ЭВМ и решения задач на нем с помощью программ, написанных на специальном командном языке.

Программные документы — документы, содержащие сведения, необходимые для разработки, изготовления, сопровождения и эксплуатации программ.

Программные закладки — программы-шпионы, которые могут вносить произвольные искажения в коды программ, переносить информацию из одних областей памяти в другие, искажать выводимую на внешние устройства информацию.

ПО (программный продукт) — «набор компьютерных программ, процедур и, возможно, связанной с ними документации и данных». Это определение, данное в международном стандарте ISO/IEC 12207:1995.

Профессиональное программирование — деятельность, направленная на получение реального программного продукта.

Ревизоры — антивирусные программы (например, Adinf и др.) подсчитывающие контрольные суммы файлов, сохраняющие эти суммы, длины файлов, даты их изменений и др. в антивирусной базе данных. При новом запуске данные базы сверяются с реальными, расхождения обнаруживаются как вирус.

Редактирующие программы — программы, предоставляющие различные возможности для изменений в тексте исходной программы.

Редактор связей — программа, выполняющая подключение (вставку) к объектному коду (откомпилированной программе) необходимых процедур и функций из библиотеки стандартных подпрограмм.

Реляционная модель БД (relation — отношение) — модель описания объектов, входящих в предметную область как совокупности взаимосвязанных атрибутов, т. е. свойств, находящихся в отношениях, построенных по принципу «один ко многим». Реляционная БД представляется чаще всего в виде взаимосвязанных таблиц.

Семантика языка программирования — система правил однозначного толкования отдельных языковых конструкций, позволяющих воспроизвести выполнение процесса обработки данных.

Сетевая модель БД — модель описания объектов, входящих в предметную область как совокупности взаимосвязанных объектов и их атрибутов (свойств). Отношения построены по принципу «многие ко многим» и «многие к одному».

Си — универсальный язык программирования высокого уровня, который дал начало целой серии языков (Си с классами, С++,

Objective-C, C-talk, Complot C) и систем (C++Builder, Visual C++) программирования.

Синтаксис языка программирования — система правил, определяющих допустимые конструкции языка программирования из букв алфавита.

Система автоматического управления (САУ) — система, в которой управление полностью выполняют технические средства, без непосредственного участия человека в этом процессе.

Система команд исполнителя (СКИ) — совокупность команд, конечный перечень доступных пониманию указаний.

Система программирования — интегрированная среда, предназначенная для автоматизации процесса программирования, которая содержит трансляторы алгоритмических языков высокого уровня и инструментальную программную оболочку, включающую редактор текстов программ, библиотеки полезных стандартных программ, отладчики и другие вспомогательные программы.

Система управления базами данных (СУБД) — совокупность методов, языковых и программных средств, предназначенных для создания, ведения и использования БД многими пользователями. СУБД позволяют создавать и хранить большие массивы данных и манипулировать ими.

Системное ПО — комплекс программ, выполняющих функции организации всех частей компьютера и подключенных к нему внешних устройств, управления ими.

Системное программирование — разработка средств системного ПО и систем программирования.

Следование — базовый алгоритмический элемент, позволяющий строить линейные структуры.

Сопровождение — анализ работы программы по результатам производственных испытаний с наборами реальных данных, обучение заказчика работе с программой, консультации и устранение ошибок, выявленных в процессе производственной эксплуатации программы.

Спецификация — программный документ, содержащий состав и назначение программ и документации к ним.

Стандарт — нормативный документ, в котором установлены для всеобщего использования общие принципы, правила, характеристики, касающиеся различных видов деятельности и их результатов.

Стандартизация — деятельность по разработке и установлению как обязательных, так и рекомендуемых для выполнения требований, норм, правил, характеристик товаров надлежащего качества, приемлемой цены, комфортных и безопасных.

Стандартные программы — общепотребительные программы, разработанные в соответствии с ГОСТ. Их можно включать в состав пакета прикладных программ для решения разных задач. Библиотеки стандартных программ формируются и содержатся на магнитных носителях под определенными именами.

Структура — определенный порядок объединения элементов, составляющих систему, это множество связей между элементами системы.

Структура программы — состав разделов программы и порядок их следования и взаимосвязей. Как правило, программа включает: заголовок, описание данных, начало программы, операторы, конец программы.

Структурное программирование — поэтапное нисходящее программирование с использованием управляющих структур следования, ветвления и цикла для решения элементарных функциональных подзадач.

Структурные схемы — специальный графический язык описания алгоритмов. Схема — ориентированный граф, стрелками (или линиями) указывающий порядок исполнения команд алгоритма, а вершины (события) такого графа представлены геометрическими фигурами, которые называются символами.

Сценарий — план выполнения работы при решении задачи на ЭВМ.

Текстовый редактор — удобное встроенное средство создания и редактирования текстов программ. Окно редактора в инструментальной среде программирования предназначено для ввода и коррекции текста программ.

Тестирование (верификация) — проверка правильности (корректности) работы программы.

Технология OLE (Object Linking and Embedding) — технология связывания и внедрения объектов, позволяющая переносить (внедрять) из одних в другие файлы информацию различного вида, создавая связанные составные документы (файлы), содержащие совокупность этой информации.

Технология Two Ways Tools — технология, в которой при размещении или изменении компонента в форме окна системы программирования соответствующий программный код будет автоматически дополнен или изменен, и наоборот.

Тип данных — определяется типом значений, которые могут принимать данные объекты (константы, переменные, выражения), и множеством допустимых операций над ними.

Тип переменной — определяет, какое значение может принимать переменная и какие операции над этой переменной возможны. Наиболее простые типы: целые числа (integer), числа с дробной частью — вещественный (real), буквы, символы — символьный (char) и др.

Транслирующие программы (трансляторы) — специальные программы, позволяющие получить из исходной объектную программу. Они осуществляют автоматический перевод программ с языка высокого уровня в последовательность машинных команд. Имеется два вида транслирующих программ: компиляторы и интерпретаторы.

Уровень языка программирования — уровень абстракции, на котором формируется задача и процессы обработки данных в реальном устройстве (процессоре). Выделяют языки низкого уровня, которые понимает реальный процессор (машинные и машиноориентированные) и высокого уровня, которые понимает человек и гипотетический процессор, и требующие трансляции на машинный язык: проблемно-ориентированные, универсальные, проектирования программ и др.

Утилита — системная программа вспомогательного назначения (например, резервирования (копирования), архиватор, русификатор, оптимизации дисков, динамического сжатия дисков, ограничения доступа и др.).

Файл — это именованная совокупность данных, хранимая в памяти ЭВМ. Полное имя файла указывает наиболее точное (уникальное) обозначение, включающее информацию о его место-

нахождении (диск, цепочку каталогов, имя файла с расширением).

Файловая переменная — специально вводимый указатель файла внутри программы.

Фортран (Fortran — Formula Translation) — первый язык программирования высокого уровня.

Цикл — базовый алгоритмический элемент, позволяющий строить структуры многократного повторения.

Шифрование — процесс преобразования открытого текста с целью сделать непонятным его смысл для посторонних. Различают шифры замены (заменяется каждая буква открытого текста на символ шифрованного текста) и шифры перестановки (меняется порядок следования букв).

Экспертная система — система искусственного интеллекта (программно-аппаратный комплекс), разработанная на основе специальных знаний экспертов об определенной предметной области (информатике, математике, медицине, геологии и т. д.).

Язык гипертекстовой разметки (HTML — Hiper-Text Markup Language) — язык для программирования во Всемирной паутине — WWW (World Wide Web), язык для написания программного кода Web-страниц. Основу HTML представляет обычный текст, в который вставляются управляющие символы (тэги).

Язык описания сценариев — подвид пакетных файлов, подобен макросам, в которых объединены отдельные команды, управляющие операционной средой в соответствии со списком, который и является программой. Язык предназначен для связывания различных компонентов и приложений друг с другом, для комбинирования компонентов, набор которых создается заранее при помощи графического интерфейса пользователя, Internet или инфраструктуры разработки и технологии сценариев. Например языки Perl, Tcl, Active X, Java и др.

Язык программирования — совокупность основных символов и правил составления из них смысловых конструкций. Язык программирования имеет алфавит, грамматику, лексику (словарный запас), синтаксис (порядок следования компонент) и се-

мантику (содержание). Это формальный язык, специально создаваемый для общения человека с компьютером.

Язык проектирования программ (система визуального программирования) — язык самого высокого в настоящее время уровня абстракции, обладающий средствами визуального проектирования. Это язык описания задач, а не описания процесса обработки данных. Как правило, в основе таких систем лежат развитые языки менее высокого уровня. Например, системы *Visual Basic*, *Delphi*, *C++Builder*, *Visual C++*, базируются соответственно на языках *Basic*, *Object Pascal* и *C++*. Они имеют интегрированную среду разработки (IDE — Integrated Development Environment), включающую текстовый редактор, конструктор форм и набор других инструментов, позволяющих значительно сократить объем работ по созданию программных продуктов.

Язык структурированных запросов (*SQL* — Structured Query Language) — непроцедурный язык высокого уровня. Представляет собой субязык, используемый для создания интерфейса БД. Реализован в популярных СУБД. Стал стандартным языком запросов в БД. Включает в себя: ЯОД — язык описания данных, ЯУД — язык управления данными, ЯМД — язык манипулирования данными. В 1992 г. объявлен международным стандартом.

CASE-технология — (CASE — Computer Aided Software Engineering) — технология создания и сопровождения ПО различных систем.

Содержание

Введение	3
История развития программирования	5

Глава 1. ОСНОВНЫЕ ЭЛЕМЕНТЫ ПРОГРАММИРОВАНИЯ

1.1. Управление компьютером с помощью программ. Система команд исполнителя. Алгоритмы. Программы. Машинные коды	17
1.2. Общие понятия о составлении программы и этапы ее разработки. Разделение программы на части	26
1.3. Виды и этапы создания программных продуктов	35
1.4. Среда программирования. Редакторы. Трансляторы. Отладка. Тестирование. Сопровождение	43
1.5. Данные. Типы данных. Структуры. Хранение данных	47

Глава 2. ЯЗЫКИ ПРОГРАММИРОВАНИЯ

2.1. Классификация языков программирования и этапы их развития	57
2.2. История развития языков программирования Паскаль и Си	70
2.3. Средства описания языков программирования. Основные понятия языков программирования	74

Глава 3. ПРИНЦИПЫ СТРУКТУРНОГО ПРОГРАММИРОВАНИЯ. НАЧАЛА ПРОГРАММИРОВАНИЯ В СРЕДЕ TURBO PASCAL. СИСТЕМА DELPHI

3.1. Принципы структурного программирования. Язык Pascal. Основы программирования в среде Turbo Pascal	78
--	----

3.2. Программирование линейных задач в среде Turbo Pascal	87
3.3. Программирование задач разветвляющейся структуры в среде Turbo Pascal	91
3.4. Программирование задач циклической структуры в среде Turbo Pascal	93
3.5. Delphi — система программирования задач на языке Object Pascal	100
Глава 4. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ И АВТОМАТИЗИРОВАННЫХ СИСТЕМ	117
4.1. Программное обеспечение для вычислительной техники. Системные программные средства	120
4.2. Программное обеспечение для автоматизированных систем управления. Стандартные и профессиональные пакеты прикладных программ	148
Глава 5. ПРАВОВАЯ И ПРОГРАММНАЯ ЗАЩИТА КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ	
5.1. Правовая защита, виды и принципы защиты компьютерной информации	159
5.2. Программный подход к защите информации. Антивирусные программы	163
Глава 6. СТАНДАРТИЗАЦИЯ И ЛИЦЕНЗИРОВАНИЕ ПРОГРАММНЫХ ПРОДУКТОВ	175
6.1. Правовые акты стандартизации и сертификации программных продуктов	175
6.2. Стандартизация, сертификация и лицензирование программных продуктов	178
Литература	187
Глоссарий	189

Гвоздева Валентина Александровна

Введение в специальность программиста

Учебник

Редактор *П. Е. Шраго*
Корректор *А. А. Лебедева*
Компьютерная верстка *И. В. Кондратьевой*
Оформление серии *Р. Остроумова*

Сдано в набор 01.06.2004. Подписано в печать 02.10.2004. Формат 60х90/16.
Гарнитура «Таймс». Усл. печ. л. 13. Уч.-изд. л. 12,7.
Печать офсетная. Бумага типографская № 2. Тираж 5 000 экз.
Заказ № 10560.

ЛР № 071629 от 20.04.98
Издательский Дом «ФОРУМ»
101831, Москва — Центр, Колпачный пер., д. 9а
Тел./факс: (095) 925-32-07, 925-39-27
E-mail: forum-books@mail.ru

ЛР № 070824 от 21.01.93
Издательский Дом «ИНФРА-М»
127214, Москва, Дмитровское ш.
Тел.: (095) 485-70-18; 485-74
Факс: (095) 485-53-18.
E-mail: books@infra-m.ru
Http://www.infra-m.ru

НБ ПНУС



687128

Отпечатано в полном соответствии с качеством
предоставленных диапозитивов в ОАО «Тульская типография».
300600, г. Тула, пр. Ленина, 109.